Real Time Implicit Bulging and Volume Preservation

Niall Linden, Hugh Reynolds, Carol O' Sullivan

Image Synthesis Group
Trinity College
Dublin 2
Republic of Ireland
{niall.linden,hugh.reynolds,carol.osullivan}@cs.tcd.ie

Abstract
Implicit Surfaces are particularly suitable for the animation of deformable objects, although some problems still remain, especially in real time applications. One problem is that of volume loss when objects are squashed. This paper presents a real time solution for the preservation of volume using dynamic creation of primitives in two dimensions. We also look at the unwanted blending problem and propose a more efficient solution.

Keywords: Implicit Surfaces, Volume Preservation, Unwanted Blending, Real Time Animation.

1 Introduction

As time has passed, and the power of computers has increased dramatically, problems that could only be solved off line are now becoming real time issues. One such problem is that of animating deformable objects. These are objects whose topology can change over time and which can join and separate.

The use of Implicit Surfaces provides an elegant solution to these problems. The idea of Implicit Surfaces has been around for over 15 years and was initially described as blobby molecules [Blinn 1982]. This was soon followed by metaballs [Nishimura et al 1985] and soft objects [Wyvill et al 1986] although they now all come under the broad category of Implicit Surfaces. However this solution can lead to problems of volume control during animation if one wishes to represent non-compressible material. In this paper we propose a solution to control volume loss and address the issue of unwanted blending.

The paper is organised as follows: Section 2 gives an overview of Implicit Surfaces and discusses previous work in relation to polygonisation, volume control and animation; Section 3 proposes a new method of preventing volume loss during collisions and a new method of keeping track of objects that have split or blended. Section 4 presents the results achieved in a two dimensional setting and finally section 5 reviews the results and discusses future work and extension to three dimensions.

2 Overview of Implicit Surfaces

In this section we will describe implicit surfaces and review related work.

2.1 Implicit Surfaces

As with many computer techniques the idea of an implicit surface is a very simple one, but with many advantageous consequences for computer modelling. An Implicit Surface can be defined as all the points *P*:

$$F(P) - Iso = 0$$
$$F(P) : \Re^3 \rightarrow \Re^1$$

where F(P) is the implicit function (scalar field function), and *Iso* is the threshold value or Iso value at which an Iso surface is extracted.

Each implicit function has an associated *primitive* (e.g. a point or a line) around which it acts. For a function to be defined as an implicit

function it needs to meet only one simple criterion. It must be a decreasing continuous function. Generally implicit functions have a value of 1 at a distance r = 0 (where r is the distance from P to the primitive) and a value of zero or approximately zero at a predefined distance. Note also how *F(P)* takes 3D space and returns a 1D value. This is why it is sometimes referred to as the scalar field function.

An interesting property of the implicit model is that it allows us to consider a surface consisting of m primitives by simply adding their scalar fields [1] (fig. 1). This process is called blending and defines a method for calculating overall field values. We also consider a terminating value *T* at which the field value is zero (radius of influence). In our model, we have used [2] as our field function due to its polynomial equation and blending characteristics.

$$F(P) = \sum_{i=1}^{m} F_i(P) - Iso \qquad [1]$$

$$F(P) = \begin{cases} -\frac{4r^6}{9T^6} + \frac{17r^4}{9T^4} - \frac{22r^2}{9T^2} + 1 & r \le T \\ 0 & r \ge T \end{cases} \qquad [2]$$

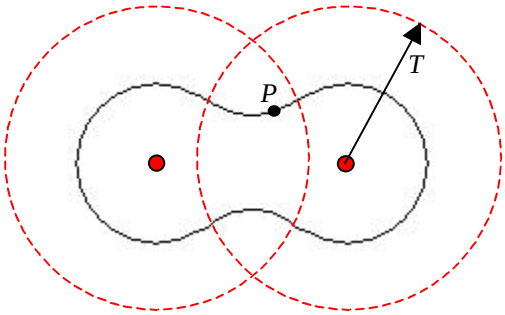


Figure 1 Two Primitives blending.

2.2 Surface Sampling and Seeds

One of the major drawbacks of implicit models is the difficulty of visual representation. The implicit representation provides a simple method for inside/outside tests but is not easily rendered at real time or even at interactive speeds. Most methods for visualisation involve some sort of spatial sampling technique (Fixed cell [Ning and Bloomenthal 1993], Surface tracking [Wyvill et al 1986] [Lorensen and Cline 1987] or Spatial subdivision techniques [Bloomenthal 1988] [Karla and Barr 1989] [Snyder 1992])

Other methods are particle based. In non-fixed particle systems, particles are scattered throughout space and then migrate to the implicit surface using the function's sign and direction gradient. Once the particles reach the surface they repel each other to produce a uniform sampling distribution [Bloomenthal and Wyvill 1990] [de Figueiredo et al 1992]. A birth and death scheme balances the distribution and eliminates the dependency of feedback terms [Witkin and Heckbert 1994].

In fixed axis models [Desbrun and Gascuel 1994], particles or seeds are used. These seeds are on fixed axes along which they migrate until they reach the surface (fig. 2). Using the field function we can determine the correct position by an inner and outer value determined by the Iso value. The value of the field can then be considered. If the field is greater than the Iso value then we continue to step out. When the value becomes less than the Iso value, we stop the seed migration. This allows seeds to move to the surface. Because in general the surface does not undergo large differences from frame to frame this allows us to have a good approximation for the next instance.

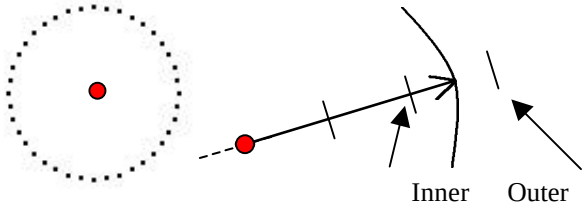


Figure 2 Seed Migration

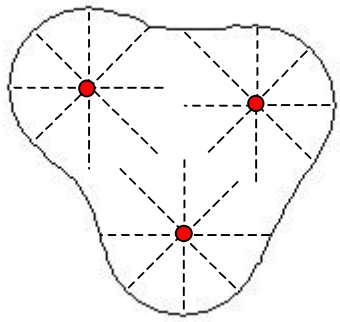
Desbrun et al [1995] ordered the seeds to produce a set of polygons that fit the surface at run time. Due to the nature of the fixed axis it is simply necessary to run through the set of polygons to display the surface at any time. The vertices of the polygons simply correspond to the seed positions. We have chosen this method for its real time performance characteristics.

2.3 Seed Invalidation

When several primitives have blended together, not all the seeds will be drawn (fig. 3). The process by which we decide not to draw seeds is called invalidation.

We invalidate a seed positioned at point P if the field value $F_i(P)$ generated by the primitive *i* is smaller than another field contribution [3]. In figure 3, we see how the seeds migrate, but if any seed finds a field produced by another (blending) primitive that exceeds the field due to its own primitive then the seed is invalidated and the next

seed is considered. We will consider this further in Section 3.



$$\exists j \neq i \mid F_j(P) > F_i(P) \qquad [3]$$

Figure 3 Seed Invalidation

2.4 Animation

Although seeds may seem to be just another method of surface representation, their strength becomes apparent in animation. When two objects collide we may wish them to blend or to form a contact surface. Gascuel [1993] proposed a method of producing piecewise contact. A negative contraction field $c_i(P)$ caused by the colliding object *i* is considered [4]. This value is than added to the blending field. This has the effect of reducing the field (as $c_i(P) < 0$) and so the seed will migrate towards the primitive centre. The final field at point P is shown in [5].

$$c_i(P) = Iso - F_j(P) \qquad [4]$$

$$F(P) = \sum_{i \in Blending} F_i(P) + \sum_{i \in Contracting} c_i(P) \qquad [5]$$

$$R(P) \propto N(P)c(P) \qquad [6]$$

We can extend this idea to one of compression before blending by simply ensuring that a value of c is exceeded before we decide that the objects are to blend. This method can be readily used by seeds in collision detection and its subsequent response. We can use seeds to sample the surface and use the contraction field as a metric for the force *R(P)* of the response of the collision [Desbrun and Gascuel 1994] where N represents the normal at the point P. Extending the idea to seeds we sum all the reaction forces on the seed to produce a primitive response vector.

3 Volume Control

In this section we discuss how volume may be preserved. Controlling the volume of an object during animation can often be very important, as the eye can be quite keen at sensing volume discrepancies.

3.1 Volume Preservation

In previous methods, attempts were made to keep the volume constant under blending of more than one primitive. In Desbrun et al [1995] local volume variation was considered by creating territories forming tetrahedral pyramids from the seeds. These were then used to create a translation that was combined with the field function to reduce the field strength of the primitive reducing volume. This idea was extended by changing the Iso value [Gesquiere et al 1999]. However, one problem is that neither of these methods produce a global method for when volume is lost (fig. 4). This is arguably a more visually unacceptable occurrence than volume increase. A more ideal solution to the eye would be one where squashing would produce bulging giving rise to volume preservation (fig. 5).

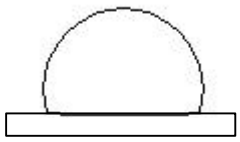


Figure 4 Volume Loss

Figure 5 Visually appealing squashing

Previous methods spread volume loss around the entire primitive and do not localise the bulging to the parts that are squashed. This does not cause the desired affect. We also wish to keep the bulging global, i.e. if another object hits a bulge it should react to it.

It is obvious to see that bulging will occur around the ends of parts that have been squashed. We can easily identify those seeds that have been squashed by considering the contraction field c on the seed. If c is not zero we know that the seed has been squashed. We can also surmise that the more the object is squashed the larger the bulge will be. To accommodate this, we introduce a metric for volume loss. Simply considering the tetrahedral

volumes would not suffice. If we have an object composed of many primitives we cannot say exactly where a seed should be in terms of distance from the primitive centre. We thus introduce the concept of an equilibrium distance.

When we update the seed so that it is on the surface, we consider its position relative to its primitive if there is no contraction field *c(P)*. We remember this as its equilibrium position, i.e. the position of the seed when no external forces are acting on it. Then when the object collides and the seed migrates towards the primitive centre, we can use the difference between its equilibrium position and its new position, *d* to approximate volume loss [7].

$$Vol\ loss \propto \sum_{s \in seeds\ /\ c_s < 0} d_s \qquad [7]$$

By having a polygon list created at run time, we can easily determine that the bulging should commence from the first seed that has polygon neighbours squashed while it remains untouched. It should be noted that both these pieces of information come for "free" with the seed method, as the information (in its raw form) is already calculated.

3.2 Bulge Creation

Now we must consider how we can create a bulge. In essence all that is required is something that can create extra volume. We decided that dynamic creation of small primitives positioned in the correct place would produce such an effect. This would have the effect of increasing the field locally, producing effects such as in fig. 5, but actually keeping the primitive information global.

The difficulty in this method is trying to place a primitive of correct strength in the correct position.
No method for inserting volume is obvious and the following issues arise (fig. 6).

1 When volume loss occurs, a bulging primitive must have a certain strength to influence the surface.
2 As d increases, the amount of volume loss increases. The difference between subsequent volume losses also increases.
3 As bulging primitive strength increases, the area it encompasses increases by a square law.

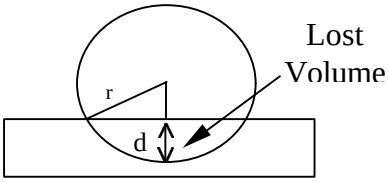


Figure 6 Volume Loss

Obviously a simple linear equation would not suffice. It would need to have components of both the primitive field function *f(x)* and the metric d. A solution of the form [8] was used to produce a visually acceptable result. This was a summation of two principle criteria. If we place the bulge primitive slightly away from the surface, we need a certain strength to influence it. This is given by the $aF(x)$ component where scalar a is dependent on the distance of the bulge primitive from the surface. The d^b term accounts for the volume loss. Due to the non-linear relationship between the volume loss metric *d* and actual volume loss a power relationship was chosen.

$$af(x) + d^b \qquad [8]$$

3.3 Inclusion of Bulge Primitive

Having created a small primitive of desired strength we are now faced with the problem of placing it in the correct place. Our first approach was to calculate the position of the bulge for the frame and insert it into the next frame. This meant that the bulging was one frame out of time, but this would not be a serious issue at real time frame rates.

A problem arose with violent collisions. Not knowing how the object was going to deform meant that occasionally bulges were placed outside the surface (fig 7). This would create primitives outside that would cause further bulges, and the system would degenerate into chaos (fig 8). Therefore a two-pass system was required. This involves first calculating the bulge positions, and then recalculating seed positions in these deformed areas. Thus at each step:

- We compute the seeds' positions and note those that are squeezed (c < 0) and sum the distances d_s to produce a metric for volume to be added.
- This is then distributed around seeds whose neighbours are squeezed.
- Small primitives are inserted at the seeds next to the last squeezed ones.
- Seed positions are recalculated in important areas where bulging has occurred.

Note the three bulges all outside and squeezing the seeds.

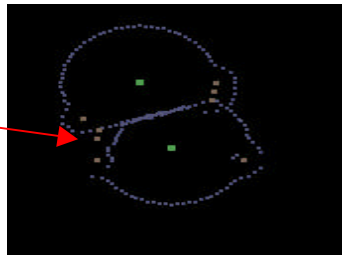


Figure 7 Bulges incorrectly placed.

Objects animated with this method can then interact with rigid or other objects with apparent volume preservation.

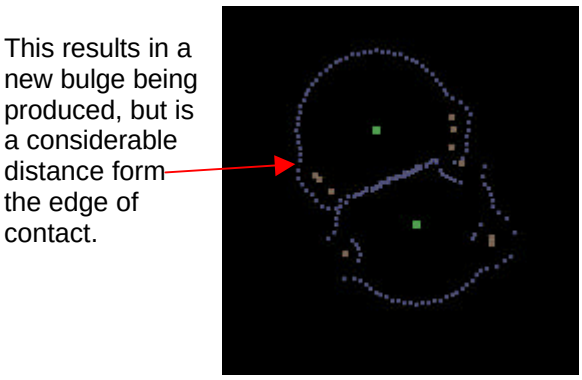


Figure 8 Bulges causing further bulges

3.4 Primitive Positioning

The positioning of the primitive can obviously produce different results. One problem is that of soft objects colliding. Due to the symmetric nature of the contact, primitives will be created opposite each other (fig. 9). One problem is that these opposite bulges will have the effect of cancelling each other out. This is hard to avoid, but by moving the primitives away from the surface we can keep some volume preservation. This is achieved by placing the bulge primitive at a fraction of the distance to the edge along the seed axis. A bulge will occur slightly above the point of contact, and then will taper off. But this bulge can deceive the eye, especially in 3D. If we are looking down, we will see the bulge but not the fact that it tapers away towards the contact surface.

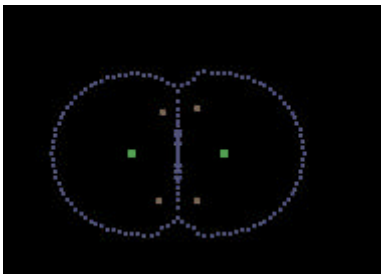


Figure 9 Opposite Bulging primitives

3.5 Unwanted Blending

The problem of unwanted blending has been around since the creation of implicit surfaces. Some objects may never blend (e.g. oil and water), and some may blend only after contact and compression (e.g. modelling clay). It is also important in animation (e.g. the arms and legs of a figure should not blend together but all the limbs should blend with the torso joints). The situation where objects split and fuse is particularly problematic.

Wyvill and Wyvill [1989] discussed the idea of a predefined blending graph, where only primitives with a connection can blend. Unfortunately this method will not work with objects splitting and blending during an animation. Desbrun and Gascuel [1995] proposed a method using radii of influence. A dynamic influence graph is created by considering transitive closure of primitive field intersections. Primitives are removed from the list when neighbours are no longer in the same influence component. This may lead to problems as the fields generally reach considerably further than the surface. Hart et al [1997] made the observation that any split or blend would lead to change in critical points, which could be used to detect topological change.

Our approach uses the idea of an influence graph that can be used to keep the connectivity of the primitive components of an object, but instead of using the radius of influence we consider seed invalidation (Section 2.2). As discussed, a seed is invalidated when it encounters a field stronger than the one from its own primitive. Thus when primitive B invalidates a seed from primitive A we receive the information that primitive A and B are joined. We can use this information to trivially construct a multi-primitive influence graph that updates itself at every frame.

This method may be extended to many primitives. If we have primitive A blending (invalidating) with primitive B and we have primitive B blending (invalidating) with primitive C then a simple algorithm can tell us that A, B and C form one single object. When an object splits, its seeds are no longer invalidated by primitives from the other object so new object detection is done automatically. Fusion is handled similarly. The main advantage is that this information has already been calculated and is known when we update the seeds position at each step. It is also virtually cost free in terms of computation time.

4 Implementation and Results

Although at first the problems of unwanted blending and volume preservation may have little in common, the link becomes apparent when we come to implement the system.

When we include the bulging primitives in the system we associate them with their ‘parent’ primitive. Each bulge primitive is a global

primitive in its own right (to ensure global contact). However blending properties must be consistent with their squashed parent. Thus we use the parents influence graph when including a bulge primitive in the system. This saves us having to create and manage new graph connections at every time frame.

4.1 Results of Volume Preservation

In our test to measure the volume preservation we conducted a test of dropping a blob on the ground. As volume measurement is difficult, Monte Carlo methods were used. A box was placed around the primitive and 20,000 points were sampled in the box. Using a simple inside/outside test against the Iso Value the results in figure 10 were obtained. This ensured volume was kept within a 7% tolerance of its original value.

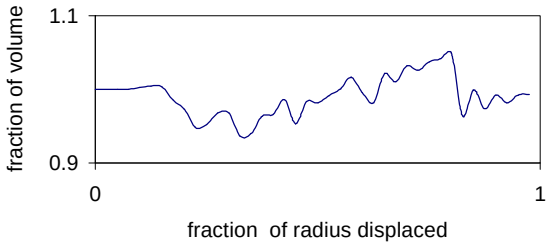


Figure 10 Volume Results

4.2 System Results

The volume preservation method outlined above was designed for use in real-time systems where there is a trade off between accuracy and speed. To test the system, we identified 3 variables.

- Number of primitives
- Volume preservation on or off
- Frequency of bulging

In all cases rendering time was not included in timing results. We consider the cases of 4, 18 and 36 primitives. Each primitive is a simple point system with 60 seeds distributed evenly around its surface. The environment is a box of size 700x700 with the radius of each circular primitive being 40 units. There is also a funnel in the top left through which the primitives can fall. We consider results for 4, 18 and 36 primitives each with 3 cases for frequency of collisions, low medium and high. In the low case, objects are blendable and are spread throughout the system. In the medium case, the objects start above the funnel (with the exception of 36 primitives where 18 start above and 18 below). In the high case, the primitives do not blend with each other, start above the funnel and collide constantly. Figure 11 shows the results of the experiments. In each case we consider the result as a % increase against the case of the objects with no volume preservation. A point to note is that in the medium cases the result drops for more primitives. This is due to the fact that the primitives form large objects and thus reduce the number of bulge primitives that have to be added. For single primitives colliding (as in the high cases) processing time increases dramatically as calculation of position, size and recalculation of seed positions takes time.

5 Analysis and Future Work

This paper has presented a method for preserving volume for use in real time animation. By using the seeds to inform us where squashing has taken place we dynamically create new primitives and place them in the system. Although it works well for contact with rigid bodies, problems still have to be addressed in collision between two deformable objects. The system is processor intensive, especially for large number of primitives. This is mainly due to the difficulty of exploiting coherency from one frame to the next during violent collisions. If one bulge primitive is placed incorrectly the results can perpetuate. A future development may be to try to differentiate between

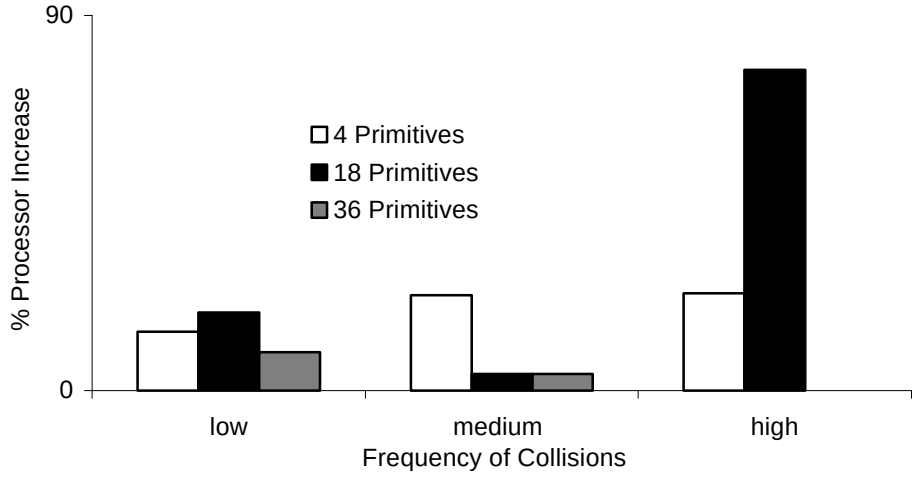


Figure 11 Processor Increase for Volumetric Bulging

collisions that we can place a bulge primitive into the next frame with certainty, and those which we cannot be sure that the bulge primitive will lie inside the surface on the next frame. Another method may be that of back-tracking. Although very useful in rigid bodies, this approach may prove to cause much more processing time and may lead to cyclic problems in deformable objects.

Currently work is underway to produce a 3D model, which involves contact surfaces as opposed to contact lines.

6 References

Blinn. J.(1982) A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235-256.

Bloomenthal. J. (1988) Polygonisation of implicit surfaces. *Computer Aided Geometric Design*, 5:341-355.

Bloomenthal J. and Wyvill. B (1990) Interactive techniques for implicit modeling. *Computer Graphics*, 24(2):109-116.

Desbrun M and Gascuel M-P (1994) Highly deformable material for animation and collision processing. In *Fifth Eurographics Workshop on Animation and Simulation*, Oslo, Norway, September 1994.

Desbrun M and Gascuel M-P (1995) Animating soft substances with implicit surfaces. *Computer Graphics*, 1995. Proceedings SIGGRAPH'95.

Desbrun M, Tsingos N, and Gascuel M P (1995). Adaptive sampling of implicit surfaces for interactive modelling and animation. In *Implicit Surfaces'95*, pages 171-186, Grenoble, France, April 1995. Proceedings of the first international workshop on Implicit Surfaces.

de Figueiredo L H ,de Miranda Gomez J, Terzopoulos D, and Velho L. (1992) Physically-based methods for polygonization of implicit surfaces. In *Graphics Interface'92*, pages 250-257, Vancouver, Canada, May 1992.

Gascuel M P (1993). An implicit formulation for precise contact modelling between flexible solids. *Computer Graphics*, 27:313-320, 1993.

Gesquiere G, Faudot D and Rigaudiere D. (1999) Volume control of equipotential implicit surfaces. Implicit Surfaces Workshop

Nishimura H, Hirai M, Kawai T, Kawata T, Shirakawa I, and Omura K (1995). Object modeling by distribution function and a method of image generation. *The Transactions of the Institute of Electronics and Communication Engineers of Japan*, J68-D(4):718-725.

Ning P and Bloomenthal J. (1993) An evaluation of implicit surface tilers. *IEEE Computer Graphics and Applications*, 13(6):33-41.

Lorensen W and Cline H. (1987) Marching cubes: a high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4):163-169, July 1987. Proceedings of SIGGRAPH'87 (Anaheim, California, July 1987).

Karla D and Barr A H. (1989) Guaranteed intersections with implicit surfaces. *Computer Graphics*, 23(3):297-306.

Snyder J M. (1992) Interval analysis for computer graphics. *Computer Graphics*, 26(2). Proceedings SIGGRAPH'92

Witkin A and Heckbert P. (1994) Using particles to sample and control implicit surfaces. Proceedings of SIGGRAPH'94.

Geoff Wyvill, Craig McPheeters, and Brian Wyvill. Data structure for soft objects. *The Visual Computer*, pages 227-234, August 1986