

Smarticles for Sampling and Rendering Implicit Models

Pauline Jepp, Brian Wyvill and Mario Costa Sousa[†]

Department of Computer Science
University of Calgary

Abstract

Particle systems are a useful way of speeding up the rendering of implicit models and creating effective visualizations of the surface. Surface features, however, can be difficult to see with traditional styles of rendering particles. Therefore particle systems are being developed that use particles to position strokes rather than draw oriented discs or other more traditional particle shapes. Detection of surface features still remains a problem related to the distribution of the particles. In this research a new technique to sample and position strokes for pen-and-ink style rendering of implicit surfaces is presented. Steering and flocking behaviours are employed to direct particles to sample and render in the same pass.

1. Introduction

Implicit surfaces are computationally expensive to render and are therefore difficult to visualize in real time. A potential solution to this problem is to use a particle system to render the surface. Earlier systems [WH94] have rendered particles as oriented discs on the iso-surface. Although this is an adequate visualization technique for simple, relatively smooth surfaces, many details of the shape and form of the surface may not be illustrated, particularly for complex surfaces. Implicit modelling systems such as the Blob-Tree [WGG99] use complex operations to create surfaces with sharp features as well as the traditionally more curved shapes.

Extensions to particle systems that position strokes in a pen-and-ink style are powerful techniques to aid in the visualization of models. Pen-and-ink strokes can be created to render feature lines and general surface strokes, which gives more information about the surface than unconnected particles alone. In the past, several particle system frameworks for implicit surfaces have been proposed. Many of these rely on particles as the rendering medium [WH94] [PS93] [MMW05] ([RRS97] also has the option to polygonize), with others extending the technique to use feature lines [TSYK01] or silhouettes [SH05], and pen-and-ink styles [BH98] [Elb98], [Aki98b] [Aki98a] [FJW*05].

Faster surface visualization comes at the cost of surface detail information. Features can easily be undetected if the particle placement is sparse, and their visualization is less effective without suitable cues. Rendering features using a pen-and-ink style provides more information about the shape and form of a surface. Details, however, can still be missed with the trade off between speed and accuracy when a particle system is used.

This research is built on the work reported in [FJW*05] with new methods of finding surface details and rendering strokes. A sampling and rendering method is presented that uses steering and flocking behaviours to find, trace and render features of an implicit model. Sampling, tracing and rendering are carried out during visualization of the surface, at interactive frame rates. Variations of the flocking and steering behaviours introduced by Craig Reynolds [Rey87] [Rey99] are used to steer flocks of smarticles (**smarticles**). Results of our system emulate traditional hand-drawn pen-and-ink illustrations.

The contribution of this work is the application of flocking and steering behaviours to both sample and render a 3D implicit model. This also improves on the previous system by providing more automatic feature extraction with less user interaction required.

[†] {pj, blob, mario}@cpsc.ucalgary.ca

2. Related Work

In [Rey87], a particle system is used to simulate group behaviour, i.e. flocks, herds or schools. The individual members of the group are modelled using particles, with the group behaviour emerging from the application of straightforward particle interaction rules. There are three main flocking behaviours: collision avoidance, or separation, where a boid will avoid collisions with its local flockmates; velocity matching, or cohesion, where the velocity of neighbours is matched; and flock centering, or alignment, where neighbouring flockmates attempt to stay together.

In [Rey99], the term *behaviour* refers to "*the improvisational and life-like actions of an autonomous character*"; this is in contrast to a scripted set of actions. Steering behaviours give autonomous characters the ability to navigate their environment in a realistic and improvisational manner; they are independent of the mechanisms of character locomotion. Combination of steering behaviours determine the total steering force that generates complex patterns. The method of combination is typically linear, and is achieved by associating weights with each of the steering direction vectors. The vehicle model is of a simple point mass approximation with a position, mass, velocity and orientation. The vehicle itself, however, is an orientable flying vehicle with a local coordinate frame. The orientation of the individual particle or vehicle determines its local coordinate basis in terms of its local origin and local forward, side and up vectors. This frame is incrementally rotated to maintain coherency in orientation of the individual vehicle throughout the simulation.

Witkin and Heckbert [WH94] presented one of the seminal particle system frameworks for implicit surfaces. Many other implicit surface particle systems are based on these principles. The attractor and repulsion forces that distribute particles around and keep them on the surface are fundamental forces that are in continued use in later systems.

In [SH05] a Witkin-Heckbert style particle system is extended to use behaviours, attributes and shaders to sense, extract and render surface information. The definition of *behaviours* in [SH05] refers to the forces applied to the particles. This includes the attractor and repulsion forces, the integration step, and the birth and death of particles as in the Witkin-Heckbert model. The main difference from the Witkin-Heckbert approach is the method of decomposition of the forces. There are other behaviours that cause particles to move towards silhouettes or singularities. These behaviours are very different from the steering and flocking behaviours presented by Reynolds. Regular floater particles are displayed as oriented particles whereas singularity particles are spheres. The feature is illustrated by a collection of spheres rather than a connected chain or stroke, as with silhouette edges. This contrasts with our system (and its predecessor in [FJW*05]) in that we use a stroke rendering paradigm.

Smart particles or *sparts* are used in [PS93] in the con-

text of virtual spray cans. The cans are filled with different types of sparts that perform different tasks. This allows the user to have control over progressive refinement of the image. Spray rendering can render arbitrary data sets in various styles. Characteristics can be assigned to the sparts such as surface seeking, volume penetrating, flow tracking, and meta-sparts. They allow the user to visualize a data set by interacting with it using a collection of specific-task sparts, or spray cans. Sparts generally need direction and don't use any type of stroke rendering or feature extraction in terms of creating a pen-and-ink style rendering. Although the flow field is similar to ours it differs in that the sparts are used to illustrate the pattern of the field rather than any surface or details in the implicit field. There is also no form of containment or constraints or flocking behaviours. The sparts are directed by the interaction with the user which contrasts to our approach in that the flock management scheme is the main coordinating method with user interaction for rendering.

RenderBots [SGS05] are autonomous agents that represent one stroke in one style. There are a number of different styles of strokes and so a number of RenderBots are used to render an image. The RenderBots can draw edges, shade an image using hatching and stippling, and use styles such as mosaic or painting. The agent space consists of a source image and possibly additional G-buffers. These are layered to create a RenderBot's *universe*. This is an image space method with the focus on a 2D environment, rather than a 3D object space as with our system. Although 3D information may be available in terms of information from G-buffers, it is not made available to the RenderBots when generating an image.

Another method that operates in image space is [SOD04]. User collaboration with an artificial ant colony progressively transforms photographs into stylized pictures. Ants are autonomous agents that use only local information and can navigate edges, fill and hatch regions, and smudge regions of the image.

The BlobTree [WGG99] paradigm has been introduced as a method of organizing implicit surface modelling in a manner that enables global and local operations to be performed in a general and intuitive fashion. It supports blending, controlled blending, bounded blending, constructive solid geometry (CSG), precise contact modelling (PCM) and spacial warping. It is a very useful tool for generating complex surfaces with arbitrary topology.

The Non-PhotoRealistic BlobTree (NPR-BT) was presented in [FJW*05]. It is a pen-and-ink style renderer that operates on any implicit surface modelling system. It was applied specifically to the BlobTree for testing complex models. A particle system based on Witkin-Herbert's is used to find interesting areas of the surface and render stylized strokes, guided by local shape features. Interesting areas include silhouette edges and feature lines such as those caused by CSG junctions. Strokes on the surface are extracted at

positions of surface particles and stylized based on a number of surface and lighting measures. Stroke extraction relies on the positioning of particles and thus features can be missed if the particle covering is not adequate. Achieving an adequate covering of particles to extract all features is the main limitation of this method. An adequate covering of particles can rarely be guaranteed, so developing other methods of steering to find and extract surface features is extremely desirable.

3. Overview

An implicit surface [BBB*97] S , composed of the set of points $\mathbf{x} = (x, y, z)$, is derived from a potential function $f(\mathbf{x})$ as follows:

$$S = \{\mathbf{x} \in \mathcal{R}^3 : f(\mathbf{x}) = iso\} \quad (1)$$

where iso is a constant value defining the iso-surface of interest. There are two key advantages to modelling with implicit surfaces. The first is derived from the fact that Eq. 1 is easily modified to define a volume (using $f(\mathbf{x}) \leq iso$ or $f(\mathbf{x}) \geq iso$), which allows for solid modelling operations to be easily applied. This also provides a useful tool for ascertaining if a queried point is inside or outside the surface. The second advantage is the ease with which smooth blends of component implicit models are achieved using simple functional compositions.

For the techniques described in this paper, the potential function is treated as a *black box*. This means that the method is general and can apply to any implicit model definition where the gradient is computable everywhere (although it does not have to be continuous). The stroke extraction methods provided in this paper rely on the vector field created by the gradient $\nabla f(\mathbf{x})$ of the implicit surface and on field-test evaluations ($f(\mathbf{x})$) for a surface with implicit value iso . The gradient of an implicit surface extends everywhere $f(\mathbf{x}) > 0$. When used exactly on the surface, the gradient is perpendicular to the surface.

The work by Reynolds in [Rey87] and [Rey99] are inspirations for the research presented in this paper. The main differences are the vehicle model, the added surface constraint and the choice of behaviours. In this research, the focus is on the visual appearance of the flock's steering behaviour (its path) and also the benefits afforded in terms of sampling space and placing particles and strokes. To this end, smarticles are based on points rather than modelling flying vehicles therefore no local coordinate frame is considered. Paths that create strokes must be constrained to lie on the surface, so a final step of correcting positions to the surface is performed.

The paper is organized as follows: first smarticle and flock initialization is described followed by their dynamics. The method of sampling the object space is outlined and then the description of rendering the strokes using the flocking and steering behaviours. The flock management scheme is then presented and is followed by conclusions and future work.

4. Smarticle and flock initialization

A smarticle is an extension of a particle, based on the model presented in physically based simulations [AWK97]. With the addition of steering behaviours [Rey87], the particles are somewhat smarter than those used in a basic particle system, thereby giving the name **smart particles**, or **smarticles**. Groups of smarticles, or **flocks**, have associated flocking behaviours [Rey99].

4.1. The particle system

In our particle system, based on [FJW*05], a pre-processing step uses random rays to initialize a small, pre-defined number of particles on the surface. They are then repositioned using a few iterations of the attractor/repulsion method [WH94]. The particles need not move continuously throughout the simulation, so movement of this type only occurs during initialization or when new particles are added.

The initial particle placement and relaxation can identify surface feature outlines [FJW*05]; these features are rendered using a chain of particles' positions to create a stroke.

4.2. Initial positions

Initial particles are the surface and feature particles described above; it is from these that smarticle's initial positions are found. The initial surface particles provide enough coverage to start the flock management scheme. They are used to spawn flocks which subsequently sample the space more thoroughly and place particles and strokes.

Individual smarticles can be spawned precisely at one of these positions, or at a point near by. Flocks of smarticles are generally spawned in a region around an initial particle's position. These are calculated as the particle's position plus some small random displacement.

4.3. Voxels

The object space is organized with a voxel grid; voxels may contain surface intersections or be wholly inside or outside the surface. As the implicit field function extends beyond the iso-surface that is rendered, voxels may be external to the surface whilst still having positive, non-zero field function values. In our system the iso-surface is usually defined to be $f(\mathbf{x}) = 0.5$ from Equation 1.

The voxels containing particles from the initial pre-processing step are flagged as containing a surface intersection and a reference is stored in the *surfaceVoxels* list. In the case of larger models, not all of the surface voxels are guaranteed to be identified by this step. This is because either the particles cover only part of the surface or the covering of particles is sparse. The unidentified voxels can be found through flock neighbourhoods and movement.

5. Smarticle and Flock dynamics

Smarticles and flocks have steering behaviours which direct the way in which they navigate the space. Some flocks are assigned the task of sampling the object space to find parts of the surface and features on the model; others are used to position strokes on the iso-surface. Smarticles can, therefore, be categorized as to whether they are constrained to the surface or not. Whatever the flock's objective, the same basic steering and flocking behaviours are used. Surface smarticles are constrained to lie on the iso-surface using the attractor force (from the basic particle system) that corrects the position to lie on the surface in the opposite direction of the gradient:

$$\vec{F}_i = (f(\mathbf{x}_i) - iso)\nabla f(\mathbf{x}_i) \quad (2)$$

As in [Rey99] a steering force is calculated from the behaviours and applied to each individual smarticle's acceleration. The physics simulation is based on forward Euler integration. Flocking behaviours act on the group of smarticles and are calculated after the steering behaviours for each individual have been found.

5.1. The steering behaviours

Variations of the Reynold's [Rey99] wander, path following, seek and containment behaviours have been implemented as methods of determining a smarticle's steering force and thus its path. Different behaviours can be blended into the calculation of the steering vector for different effects.

The **wander** behaviour can be used for a particle to wander around in space or on the surface of the object. A random vector is traditionally used with some constraints to ensure the path is not too erratic [Rey99]. This involves making small, incremental, random changes to the path. The sum of the scaled velocity vector and the steering direction keep the steering force within a certain region around the velocity vector.

If there is no random component to the path then **flow field following** is the mechanism by which the steering behaviour is calculated. To implement this behaviour, typically, either the local principal directions of curvature or the contour direction are used to calculate the flow field vectors. Field vectors can also take the form of axial direction vectors, which creates (globally oriented) horizontal and vertical steering forces. Combinations of any of these can also be used to create the field flow. Changes in the flow field are typically small or smooth enough that no constraints are necessary to ensure a smooth path. The only constraint applied should be if the path is to be constrained to the iso-surface. Surface discontinuities, sources or sinks are the main areas where the flow field changes may not produce a smooth (or desirable) path.

Containment is used to constrain a flock or individual smarticle to stay within a particular region. In the context of

this research a region is defined by the curvature of the surface; a volume (possibly a voxel); or distance from a CSG or silhouette outline. At any timestep, the current position of a smarticle can be used to evaluate surface properties for containment. If there is a change in the relevant property then the path is terminated without inclusion of the current position. Where the smarticle is constrained to lie in a convex or concave region, the curvature of the surface is calculated; if the sign changes (indicating either a convex or concave region), the path is terminated (Fig. 2). If a discontinuity in the field is detected the path is similarly terminated. If the path is constrained by the curvature of the path then the surface normal at the smarticle's current position is compared with the normal at its initial position. If the angle between the two vectors is greater than desired the path is terminated. Similarly, if the containment refers to a region around a feature line, the distance between the closest point on the feature line and the current position is the measure; where the distance is measured by the lengths of the steps; the number of steps can also be used. The choice of containment is generally left to the user.

5.2. The flocking behaviours

The three flocking behaviours are implemented in the same way as in [Rey87]. Where smarticles are constrained to lie on the surface their positions must subsequently be corrected to lie on the surface (as with the general surface correction method, see Equation 2).

6. Sampling the object space

Steering behaviours are used to sample the object space to find the surface and its features. Previous methods have relied on querying the formal definition of the model or on the attractor/repulsion method moving particles on the surface to identify features. Through smarticle and flock movement other voxels that contain surface intersections are identified. The voxel is queried and if it has not been sufficiently sampled then it is investigated further.

6.1. Smarticle and flock sampling

The sampling method for all smarticles is the same. As each smarticle moves it evaluates the field function to determine if it is smoothly varying, or has some potential discontinuities or small scale details.

At each step the smarticle evaluates the field function and compares this to the value at its previous position; if the change is greater than what would be expected from a smoothly blended surface then the region is investigated further (Fig. 1). This involves finding the normals at each of these points and examining the angle between them. If the angle is greater than that expected from a smoothly blended surface the points identify a discontinuity or a small scale

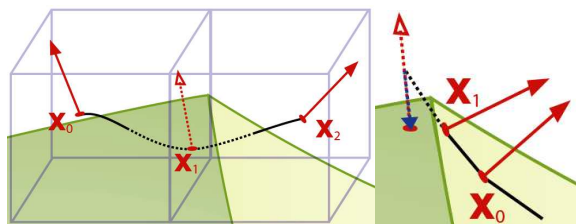


Figure 1: *Left:* An unconstrained smarticle: $f(X_0) < iso$ and $f(X_1) > iso$, indicating a surface intersection. The angle between the normals at steps $f(X_1)$ and $f(X_2)$ show that a discontinuity has been identified. $f(X_0)$ and $f(X_1)$ are in a listed surface voxel, as the path moves to $f(X_2)$, the adjacent voxel is added to the list. **Right:** The smarticle's step is corrected to lie on the surface (blue). The final position (red dashed) is rejected as it takes the path out of a concave region of the surface.

detail in the field. These points are then corrected to lie on the iso-surface and the region is evaluated in the same manner to determine if the feature is also present on the surface. If so, the points are classified as straddling a discontinuity or small scale detail in the field. Particles are created at these points and added to the relevant particle lists, i.e. the surface and straddle particles lists.

The surface normals can also be used to identify if a path has crossed a silhouette outline. The dot products of each of the normals with the view vector are examined; if one is positive and the other negative then a silhouette outline has been crossed. These points can be used to identify a silhouette edge on the surface by being added to the list of silhouette points. Silhouette and CSG points identified in this way can then be used to trace features on the surface.

6.2. Voxel sampling

Voxels that contain surface intersections are identified from the initial particle placement, through neighbouring surface voxels, or through flock or smarticle movement.

As mentioned above, particle systems have the disadvantage of not guaranteeing that every feature of the surface will be identified and rendered. The rendition will only be as detailed as the size of the sampling. Voxel sampling, as it is used here, can easily miss small details of the surface and situations can arise where the surface intersection does not encompass a voxel corner or sample point. Voxel sampling, in the context of this research, is mainly used as a fast method of identification of possible surface voxels. It is not intended as a guaranteed, robust surface intersection test.

The initial particles provide the initial *surfaceVoxel* list. The list is updated by other methods which first identify *potential* surface voxels. These often require further investigation to determine if they contain surface intersections.

When identification is through neighbouring voxels, potential surface voxels are identified where they share a face with a listed surface voxel.

The first pass of sampling a voxel that has no information available is to evaluate the field function at the corners and centre. Field function values both above and below the iso-surface value immediately identify a voxel containing a surface intersection. Where this test is inconclusive, the voxel is subdivided and further samples evaluated at the corners of each of the subvoxels.

Flocks also sample the voxel space whilst being directed by the relevant behaviours. The information from the unconstrained flock path steps can tell us if there is an intersection if it has field function values both above and below the iso value (Fig. 1). If the flock information gives us no such insight then the voxel should be sampled as if no information is available.

Further investigation is carried out on a voxel if it has not reached an acceptable level of sampling. An acceptable level is determined from information such as the number of smarticles in the voxel, the subvolume that the flock or smarticles have sampled, the estimated variation in the field function values and the detail required by the user.

Where a new flock is assigned the task to further investigate a region it can be spawned at a random position within the voxel or in a particular subvoxel.

7. Rendering the strokes

Smarticles also place strokes for rendering the surface. Strokes are placed on the surface using the same flocking and steering behaviours as for space sampling whilst also being constrained to lie on the iso-surface. Depending on the type of stroke required, initial positions for the smarticles are determined. Next, the relevant steering and flocking behaviours are used to direct the path of the strokes. The strokes are terminated according to some end conditions,

Individual or combined strokes are powerful rendering techniques to illustrate shape and form. There are a number of commonly used stroke positions and directions that we emulate here.

Smarticles using the steering behaviours alone achieve a more regular appearance in the strokes they render. Using the flocking behaviours adds a certain random appearance to the strokes. Changing the weights of the flocking behaviours allows the user to choose different styles.

7.1. Choosing a stroke

The choice of stroke will depend on both the illustrators preference and what is appropriate for the region of the surface. This decision will help identify whether to use a single smarticle to create an individual stroke or to use a flock to create

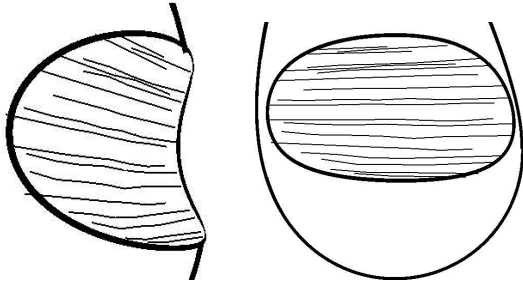


Figure 2: Concave surface strokes started from a CSG feature continue until the curvature of the surface changes. As the CSG feature stroke is an approximation of the actual discontinuity, and the field is discontinuous it is difficult to guarantee that the strokes faithfully remain within the concave region.

a group of strokes. Individual smarticles can also trace out strokes that are rendered as if they were a group. Strokes can also be layered to create different effects, such as hatching. The ordering and layering is user controlled.

7.1.1. Initiating strokes

Existing surface particles provide initial positions and surface information, such as curvature information. This information can be used, for example, to create strokes specifically on convex or concave areas, which is useful to contrast different parts of the surface.

Alternatively, initial positions can be taken from existing stroke chains that depict feature lines, such as silhouette or CSG outlines.

7.1.2. Single strokes (or paths)

A smarticle can be used to create an individual line of emphasis or a single stroke. The starting position is obtained from one of the listed surface particles. Individual smarticles are also used if a path is required, along which strokes will subsequently be drawn by flocks (much like using a feature chain). The user can decide at what angle (to the path) the strokes are drawn.

7.1.3. Groups of strokes

Flocks are used mainly for shading regions or drawing sketchy outlines (where there is more than one line) of features or regions of the surface. An example of the former (Fig. 2) would be where an illustrator requires a concave region of the surface to be shaded. In this situation, the initial positions of the smarticles in the flock are based on the position of a particle that is listed as lying in the concave region of the surface. The flow field following behaviour is then used to direct the flock members and ultimately leave strokes

on the surface. The direction of the flow field is chosen by the user.

When strokes are required to shade areas close to a feature line a number of the positions identified from the silhouette chain are used as the starting positions for the smarticles. The density of the strokes is user defined and is achieved by choosing an incremental step size with which to index the positions on the chain. The most dense shading would involve flock members obtaining their starting positions from consecutive chain elements; whereas a lighter shading would get starting positions from every n^{th} element on the chain.

7.2. Directing the path of the strokes

The steering behaviours for the individual smarticles are applied in the form of flow field following. When flocks are used in this way, there are certain considerations of the level of influence each flocking behaviour has on the member smarticles. If the lines originating from a silhouette outline are required to be almost parallel to one another, then the weighting of the cohesion behaviour would be less than that for the separation and alignment behaviours. If the cohesion weight is too strong then the strokes will ultimately converge. The required density of the strokes and the rendering will dictate the weighting for the separation behaviour. In a region of dense strokes it may be more acceptable to have strokes closer together. Weighting of each of these behaviours can be defined by the user, although preset values are initially assigned; this can be of great benefit as a small amount of adjustment is usually all that is necessary, rather than complete re-adjustment of the weights.

7.3. Terminating the strokes

The decision to terminate a stroke can be based on curvature of the surface or distance from the feature or path. Curvature can be used to generate end conditions in two ways. First, curvature information identifies when either the smarticle has moved from a concave to a convex region of the surface (or vice versa) (Fig. 1). Secondly, local curvature is measured using surface normals at each step in the stroke. The stroke continues while the angle between the normal at the first position and the normal at the current position is within a user specified range. Distance measurements are also evaluated in one of two ways. Either stroke length is measured as the cumulative length of each step in the path, or the number or steps taken. The initial values are related to the width of a voxel, in object space terms, although the user can also alter stroke length (Fig. 2).

7.4. Rendering strokes

A user is free to layer the strokes in any manner s/he wishes, thereby creating patterns such as cross hatching and rendering different regions in different styles and tones (also using the density of the strokes) (Fig. 3).

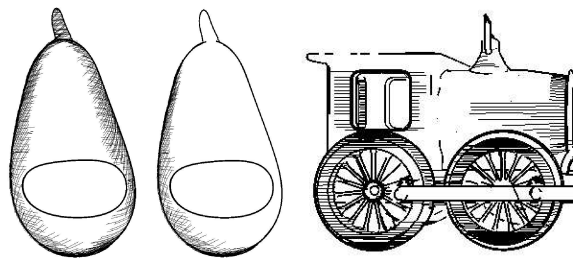


Figure 3: *Left: Layering of first principal direction and diagonal strokes with initial positions provided by the silhouette; Centre: with lighting calculations to determine the visible strokes. Right: The cab of a train using flow field flocking using horizontal stokes and no flocking behaviour.*

Although many strokes are calculated, not all of them will necessarily be rendered. Lighting plays an important part in the choice of strokes to render. If the strokes are to illustrate shade then a simple occlusion test is performed to determine if the strokes are in a shaded region of the surface. This occlusion test can not only determine if an entire stroke is displayed but also it may shorten an existing stroke if only part of the stroke is in a shaded region. The system has an upper and lower threshold measurement that creates a penumbra inside which strokes are randomly selected to be rendered, [FJW*05].

8. Flock management

A flock management scheme is necessary to coordinate flocks of smarticles and how they sample space and place particles and strokes. It is the mechanism by which the above tasks are assigned and coordinated. It controls the data structures and the general flow of the program.

The decision as to whether a flock should, for example, remain in the voxel inside which it was born depends on the task of the flock. Flocks can be assigned one of several different tasks. These are: sampling the space inside the current voxel; sampling the local neighbourhood (regardless of voxel); evaluating certain characteristics of the field function; and tracing specific regions or features of the surface.

Evaluation of field details or features is also carried out by the flock manager. It examines the steps in a smarticle's path to determine if the step has crossed a silhouette edge, or feature or discontinuity in the field. The manager also coordinates the investigation of the potential surface voxels identified by either smarticle movement or neighbouring voxels.

The tally and measurement of levels of sampling as well as identification of undersampled voxels are also flock manager tasks. This is often used to subsequently identify a voxel's undersampled subvoxels when assigning initial positions to unconstrained flock members.

A voxel has an indication of estimated and completed work. This identifies whether the voxel has been sufficiently sampled or which of its subvoxels require further investigation. The estimate is updated as more information becomes available. Such information allows the manager to identify that a particular subvoxel is less sampled than its neighbours and direct a flock to sample that region.

9. Conclusion and future work

We have described techniques to extend a particle system to include a flock management scheme. The flock management offers an alternative method of finding and rendering surface details. Steering and flocking behaviours are used to guide smarticles that sample the implicit field, position new particles and render pen-and-ink strokes. Particle systems take time to distribute themselves over the surface and find features. This method can speed up feature identification (Fig. 4). The same model is rendered using both the previous and current version of the NPR-BT [FJW*05]. The rendering is at an early stage (ie not all details have yet been found or rendered), but it is clear that the flocking and steering behaviours have identified more features than the basic particle system. Even with a better method of finding features, there still exists the trade off between speed and accuracy. Our method is not guaranteed to find all surface details.

The sampling approach shows a great deal of promise and will be developed further to include some learning and co-operation concepts that should expedite the identification of features of the field. This is with an aim of providing real-time interaction with complex implicit models.

The results from using the flocking behaviours have more free-form appearance, whereas using the steering behaviours alone is more reminiscent of formal technical illustrations. The control of this is through selection of weights for the flocking behaviours. A useful direction for this to be developed would be to have a more intuitive user interface. Using visual examples would be an effective way to help control both stroke direction, density, regularity and appearance. The strokes themselves rendered here are simple OpenGL lines, it would also greatly enhance the appearance of the strokes to use the paths of the flocks to position higher quality strokes. Future work will include using techniques from high quality real-time stroke rendering research to enhance the visual appearance of the strokes themselves.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments and suggestions. We would also like to thank Dr. Jörg Denzinger and Ian Burleigh for their suggestions. This research was supported by Discovery Grants from the National Science and Engineering Research Council of Canada.

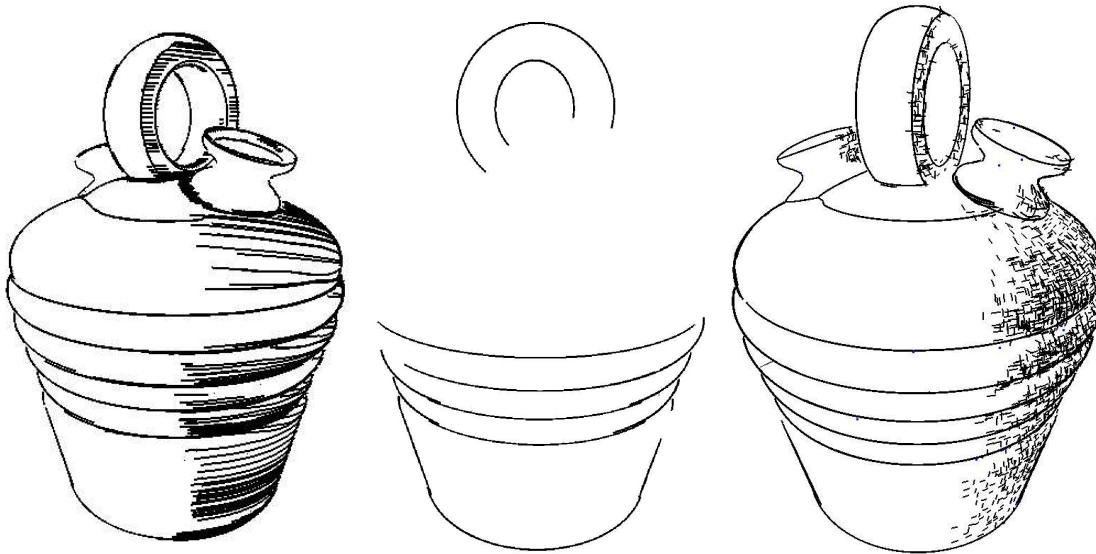


Figure 4: *Left:* The jug with flocking strokes in a sketching style. *Centre:* The jug left for the same time without the flocking behaviours and strokes. *Right:* jug rendered with the previous method using short strokes [FJW*05].

References

- [Ak198a] AKLEMAN E.: Implicit painting of csg solids. In *Proc. of CSG '98, Set-Theoretic Solid Modelling: Techniques and Applications* (1998), pp. 99–113. 1
- [Ak198b] AKLEMAN E.: Implicit surface painting. In *Proc. of Implicit Surfaces '98* (1998), pp. 63–68. 1
- [AWK97] ANDREW WITKIN D. B., KASS M.: An introduction to physically based modeling, 1997. 3
- [BBB*97] BLOOMENTHAL J., BAJAJ C., BLINN J., CANI-GASCUEL M., ROCKWOOD A., WYVILL B., WYVILL G.: *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997. ISBN: 155860233X. 3
- [BH98] BREMER D., HUGHES J.: Rapid approximate silhouette rendering of implicit surfaces, 1998. 1
- [Elb98] ELBER G.: Line art illustrations of parametric and implicit forms. In *IEEE Transactions on Visualization and Computer Graphics, Vol. 4, No. 1* (1998), pp. 71–81. 1
- [FJW*05] FOSTER K., JEPP P., WYVILL B., SOUSA M. C., GALBRAITH C., JORGE J. A.: Pen-and-Ink for BlobTree Implicit Models. *EUROGRAPHICS2005* 24, 3 (Sept. 2005), 267–276. 1, 2, 3, 7, 8
- [MMW05] M.D. MEYER P. G., WHITAKER R.: Robust particle systems for curvature dependent sampling of implicit surfaces. In *Shape Modeling and Applications* (2005), pp. 124–133. 1
- [PS93] PANG A., SMITH K.: Spray rendering: Visualization using smart particles. pp. 283–290. 1, 2
- [Rey87] REYNOLDS C. W.: Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics* 21, 4 (1987), 25–34. 1, 3, 4
- [Rey99] REYNOLDS C.: Steering behaviors for autonomous characters, 1999. 1, 2, 3, 4
- [RRS97] RÄUSCH A., RUHL M., SAUPE D.: Interactive visualization of implicit surfaces with singularities, 1997. 1
- [SGS05] SCHLECHTWEG S., GERMER T., STROTHOTTE T.: RenderBots—Multi Agent Systems for Direct Image. *j-CGF* 24, 2 (June 2005), 137–148. 2
- [SH05] SU W. Y., HART J. C.: A programmable particle system framework for shape modelling. In *Shape Modeling and Applications* (2005), pp. 114–123. 1, 2
- [SOD04] SEMET Y., O'REILLY U.-M., DURAND F.: An interactive artificial ant approach to non-photorealistic rendering, 2004. 2
- [TSYK01] TANAKA S., SHIBATA A., YAMAMOTO H., KOTSURU H.: Generalized stochastic sampling method for visualization and investigation of implicit surfaces. *Comput. Graph. Forum* 20, 3 (2001). 1
- [WGG99] WYVILL B., GALIN E., GUY A.: Extending The CSG Tree. Warping, Blending and Boolean Operations in an Implicit Surface Modeling System. *Computer Graphics Forum* 18, 2 (June 1999), 149–158. 1, 2
- [WH94] WITKIN A. P., HECKBERT P. S.: Using particles to sample and control implicit surfaces. *Computer Graphics* 28, Annual Conference Series (1994), 269–277. 1, 2, 3