Computational Aesthetics in Graphics, Visualization, and Imaging (2009)
O. Deussen and P. Hall (Editors)

Style Nodes and Repolygonization for Hierarchical Tree-Based Implicit Surface Modelling

Pauline Jepp[1], Bruno Araujo[1], Joaquim Jorge[1], Brian Wyvill[2], and Mario Costa Sousa[3]

[1] INESC-ID, Lisboa, Portugal
[2] University of Victoria, Canada
[3] University of Calgary, Canada

Abstract

In this paper we present an extension to a hierarchical tree based implicit surface modelling system that includes interactively controlling style and appearance, and also creating a more accurate curvature based polygonal approximation. Multiple styles can be layered and applied to objects so that they are guided by local geometry although not strictly bound by it. To achieve this a new node, the Style Unary Node, is added to the ShapeShop BlobTree, which creates a style blending region inspired by primitive field blending. As visualization of implicit surfaces in interactive environments is often based on polygonization a more accurate curvature based polygonisation algorithm is also presented.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]:

1. Introduction

Techniques used for scientific illustration often combine several rendering styles to create an image where particular aspects of a model or scene are emphasized. Altering a model's appearance includes changing: the colour; the style of rendering eg pen-and-ink, wireframe, shading styles; and transparency.

Illustration style choices are often *guided* by geometry, but not necessarily *bound* by it. There are two common approaches to illustrating a model with multiple styles. One approach is to associate appearance attributes with parts of a model (or scene), for example using shapes, primitives or vertices. Alternatively, a viewing lens can be used to highlight a region of interest where there is no connection to the underlying geometry.

In skeletally based **I**mplicit **S**urface **M**odelling (ISM) systems primitives (individuals or groups) naturally define parts of an object, using these boundaries can be very useful in attributing appearance or style. Associating appearances that are strictly limited to primitives, however, does not offer any complexity in terms of blending multiple styles over regions of a model. Lenses are very effective at blending styles and avoiding strict object or geometric boundaries. But because there is no connection to the underlying shape, benefits of using the geometry are lost.

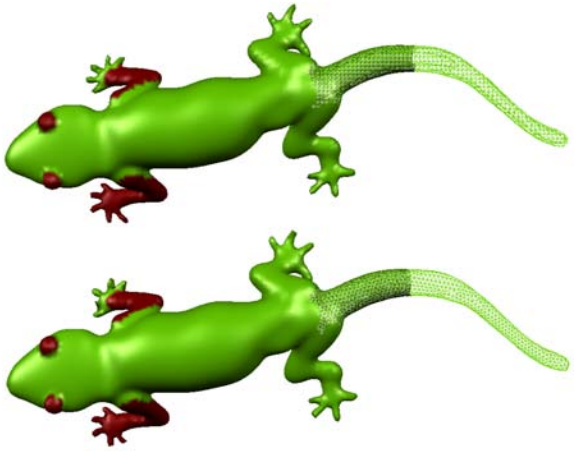

Figure 1: *The Gecko model with SUNs.* ***Top:*** *Continuation algorithm.* ***Bottom:*** *Curvature based polygonization.*

In this research a method is presented that combines the

DOI: 10.2312/COMPAESTH/COMPAESTH09/041-048

strengths of both primitive-assigned attributes and lenses. The method is applied to ShapeShop [SWG05] a sketch based, tree structured ISM system, where complex objects are constructed from simpler primitives.

Implicit surfaces provide a solution of choice for volume modelling applications. This is due to the representational compactness and the rich set of editing operators that can easily be defined.

Multiple styles can be combined (or layered) and applied so as not to be strictly limited to shape boundaries, rather they are guided by them. Using Implicit Surfaces allows styles to be blended between regions in much the same way that primitives can be blended. A new node has therefore been added to an ISM tree. The **S**tyle **U**nary **N**ode (SUN) is only used for altering the appearance of of an object and does not affect its geometry. This node facilitates assignment of styles to regions of interest whilst not limiting the styles to traditional primitive boundaries. It also allows layering and blending of multiple styles and can be moved interactively anywhere in the modelling tree.

A secondary contribution of the work presented in this paper is a curvature based polygonization algorithm. Interactive visualization of implicit surfaces generally use a spatial subdivision, continuation algorithm for speed. Although is is a fast method the results are not very smooth when viewed with resolutions used in interactive ISM systems. We present a curvature based algorithm that produces more accurate approximations of the surface in acceptable times.

The remainder of this paper is organised as follows: in Section 2 the related work is discussed. Section 3 contains the details about the implementation of the Style Unary Node. The polygonisation method for better approximation of the surface is described in Section 4. Results are presented in Section 5 and Conclusions and Future Work in Section 6.

2. Related Work

An *Implicit surface* [BBB*97] S, is composed of the set of points derived from a scalar field function $f(p)$ as follows:

$$S = \{p \in \Re^3 : f(p) = iso\} \quad (1)$$

where *iso* is a constant value defining the iso-surface and $p = (x,y,z)$. Functions $f()$ are called *Fields*, and specific values of $f(p)$ are referred to as *Field Values*. Differential geometry is used to ascertain geometric attributes, for example, the gradient of $f(p)$ defines the normal vector of the surface and the second order derivative can be used to extract shape characteristics such as curvature.

Implicit Surface Modelling (ISM) systems often employ a tree-based data structure for model representation where implicit primitives can be represented using geometric primitives and modelling operations. In [WGG99], Wyvill et al. propose a structure called the BlobTree, which is a hierarchical tree-based method that allows arbitrary compositions of models using blending, warping and boolean operations from skeletally based primitives. Pasko [PASS95] describes a function representation modelling system known as FRep. In [AGCA06], Allègre et al. present a hybrid modelling framework called the HybridTree, which is also an extended CSG tree. The HybridTree uses implicit models and polygonal meshes for editing operations.

Sketch based ISM systems allow a user to create implicit models directly from user's strokes. Teddy [IMT07] was the first free form sketch based modelling concept (although it used polygonal models). Karpenko et al. [KHR02] developed the concept further to achieve a smoother appearance by using variational implicit surfaces. Araujo and Jorge [AJ03] improved the method presented by Karpenko by improving the identificaiton of details. Convolution surfaces are used in ConvMo [TZF04]. And in FreeFormSketch [AJ05b] Araujo presents a rich set of operators mixing variational implicit surfaces and multi partition of unity [OBA*03].

In ShapeShop [SWSJ06], Schmidt et al. included hierarchical implicit volume models, specifically BlobTrees, to allow more complex objects to be created. The hierarchy can be viewed as a construction history and individual sketched components can be non-linearly edited. ShapeShop has previously only displayed objects in solid colours, using pen-and-ink stipple and strokes [SIW06], or interactive decal compositing [SGW06].

Several non-photorealistic rendering techniques have been applied to implicit surfaces to enhance their appearance. Rendering features such as the silhouette outlines [BH98] improves the expressiveness of a model. Foster et al. [FJW*05] present a pen-and-ink rendering system using a Witkin-Heckbert based particle system [WH94] to trace the silhouette (in a similar manner to Bremmer and Hughes [BH98]) and also track discontinuities or small scale details of the surface. Differential geometry can be used to identify ridges and valleys of an implicit surface [BPK98, OBS04] and render them using a polygonal approximation.

Effects are generally applied to models globally without allowing the user any fine control over the appearance and combining styles. Changing the appearance of an object can be achieved by using lens-type filters that have no connection to underlying geometry or by associating style properties with an object. Bier et al. presented MagicLenses™ [BSP*94] as user interface tools based on the principle of a magnifying glass, which used filters to modify the appearance of application objects. This technique has been extended to viewing 3D objects [VCWP96, PC03, RH04].

Cole et al. [CDF*06] use a technique similar to a lens that is applied to 3D models for architecture. A "stylized focus" effect (circular, spherical or planar) is created where a user

specified region of interest is automatically rendered to have the focus of attention.

Techniques that associate styles with defining geometry are frequently aimed at visualizing 3D scan data. Ebert and Rheingans [ER00] combined volume rendering and non-photorealistic rendering to produce a wide range of illustration styles. Hauser et al. [HMiBG01] present a volume rendering technique that allows the user to select rendering styles for different subsets of 3D data. VolumeShop [BVG05] uses volume data in a dynamic 3D illustration environment that combines artistic styles with visualization techniques. In [BG07] Bruckner and Gröller introduce *Style Transfer Functions* for segmented volume data. Styles are captured from existing artworks and a single image combines different shading styles.

Visualization of implicit surfaces requires the sampling of a scalar field to create an image. Approximating implicit surfaces using polygonal meshes (polygonizing) is used for real-time visualization using commodity hardware. Cell partitioning techniques are the most popular method, where a mesh is created based on space subdivision. Wyvill et al. [WMW86] proposed a polygonization process by subdividing the space into cubic cells. This is also the basis of the Marching Cubes algorithm [LC87]. Surface tracking approaches such as Hilton's [AHASJI96] Marching Triangles follow the surface and minimises poor edge ratios and sampling errors. Hartman [E.98] proposed a similar approach based on point-set expansion. Cermak [CS02] proposed a variant of Marching Triangles named Edge Spinning.

Adaptive polygonization captures local shape characteristics without increasing the global resolution. Both Akkouche [GA01] and Cermak [CS04] present extended marching triangles. Similar methods using additional local subdivisions steps were decribed by both Paiva [PLLdF06] and Bouthors [BN07]. Karkanis [KS01] uses a curvature radius heuristic to generate an adaptive. Araujo's [AJ05b] curvature dependent polygonization uses mean and gaussian curvature in heuristics for triangle size definition. The approach, however, was only supported by smooth variational implicit surfaces.

Previous sketch-based ISM systems do not concentrate on providing flexibility for assigning styles. One of the motivations for the work presented in this paper is to identify a method of representing style choices in a coherent and flexible manner. The ShapeShop BlobTree is ideal for this task due to the sketch based metaphor and a hierarchical tree structure. This provides an ideal base with which to enrich an ISM system (capable of producing complex models) to also include complexity in terms of appearance. As style visualization in interactive ISM systems is generally based on polygonization, a secondary aim of this work is to provide the user with a better approximation to the surface. Curvature should be considered for a good approximation to an implicit surface and also to accurately reproduce shape features.

3. Style Nodes

Style **U**nary **N**odes (SUNs) are inserted anywhere in the modelling tree and affect the appearance of individual or groups of primitives. Properties are applied to either single primitives (child leaf nodes) or groups, which constitute a sub-tree of the complete modelling tree. Any number of nodes can be inserted anywhere in the tree and their appearances are layered and blended.

Techniques inspired from implicit primitive blending allow styles to extend beyond shape boundaries to affect the appearance of neighbours. Cached field values are referenced to create a smooth and user-variable transition.

In Figure 1, a gecko has been rendered using a collection of SUNs. The first is applied globally and defines that the main colour is green, this SUN is applied to the whole tree, i.e. it is inserted directly below the root node. Other SUNs are applied to the front legs and tail to achieve the changes in colour and style. Notice that the gecko's tail (a single primitive) has a SUN to specify that it should be drawn in wireframe, with a blend region where both transparent polygons and wireframe are shown.

3.1. The Tree Traversal and SUN List Creation

SUNs are internal nodes and are treated like compositions (blend, union, intersection, difference) in a tree traversal. A traditional tree traversal evaluates compositions of primitives to determine the final geometry of the object, i.e. the field value for any point in space. Style nodes are evaluated along with field evaluations. A tree traversal returns field values to be used in *shape compositions* and also any SUNs to be used in *style compositions*. In order to keep shape compositions and style compositions separate Style Nodes are implemented as unary nodes. They do not contribute to the field value they merely pass the value from their descendants through to the parent when an evaluation is made. A list of SUNs is returned in the same evaluation step to minimise computation time. SUNs store vertex field values for their descendants during the tree traversal. Recalculation is necessary only when the descendant subtree is altered.

3.2. Layering and Blending

Displaying multiple ancestors is achieved by **layering** styles. There are three types of style changes that are considered:

1. Colour
2. Polygon appearance: wireframe, smooth polygons, transparency
3. Lines: stipples, strokes, features

This does not create conflicting situations as can be seen in Figure 2 where stipples, wireframe and smooth shading are all used to illustrate a single model of a dolphin.

Blending regions are created between a child affected by a SUN and its neighbours. The appearance of polygons is

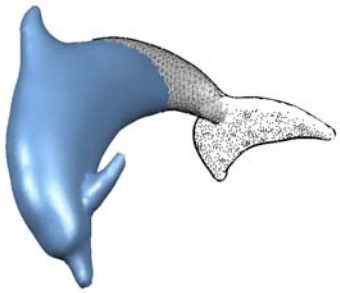

Figure 2: *The dolphin model polygonised using the curvature based method and illustrated using SUNs to identify regions with filled polygons, wireframe, stipples and silhouettes.*

blended between the two styles, for example a *blend region* between a wireframe and smooth shaded part of a surface is created and contains both the wireframe and transparent filled polygons (details follow), see Figs. 2 and 1. Stipple density is also reduced in blend regions relative to the cached contribution of the blending primitives.

In general the **blend region** between implicit primitives is defined as the volume between shapes where each primitive contribution is less than *iso* but the combined contribution is equal to *iso*. In this research the **style blend region** extends primarily over the wireframe object where the contribution from any neighbouring object is above a user variable value, although in practice $\frac{iso}{2}$ gives good results.

As mentioned in Sec. 3.1 vertex field values are stored for SUN descendent subtrees during the tree traversal and list creation. These values are used to identify triangles that lie in the blend region. To minimise computation time the field values are stored for referencing rather than recalculated.

The final object is drawn using a collection of four meshes. One for each of: shaded polygons; wireframe; mixed wireframe and filled polygons; and stipples. Triangles are evaluated for the mesh they belong to during the polygonisation process. The SUN list is referenced for each vertex to determine how triangles should be drawn, i.e. to which mesh they are assigned.

A triangle is added to either the wireframe, mixed polygonal or stipple mesh for rendering the surface. Stipples are created from polygons, but are corrected to lie on the actual implicit surface (rather than the polygonal approximation). Therefore, the stipple mesh (where present) contains a copy of all relevant triangles and the mesh is passed to the method to create surface stipples, as explained in [SIW06]. The triangles used for calculating the stipples are not drawn, as can be seen in Figs 2 and 7.

if $triangle == stipple$ **then**
  $StippleMesh.add(triangle)$
else if $triangle == WireFrame$ && $triangle \in BlendRegion$ **then**
  $MixedMesh.add(triangle)$
else if $triangle == WireFrame$ && $triangle \notin BlendRegion$ **then**
  $WireFrameMesh.add(triangle)$
else
  $PolygonMesh.add(triangle)$
end if

3.3. Family Traits

SUNs affect either the **family** or only their **children**. Family traits affect all of the descendants, which is useful for changing the appearance of a large region of the model i.e. an entire subtree, See Figure 1 where the gecko's front left leg is red, the subtree contains the blended leg and foot primitves.

Child traits affect only immediate children, which is useful for limiting the appearance of changes to restricted areas of a surface. For example, see Figure 1 the front right leg has its colour changed but the foot, its child node, has the colour of the initial SUN (directly below the root node).

Where a style node is set to only pass on its appearance traits to its immediate children, the descendants do not have this SUN added to the list of contributing nodes.

4. Curvature based polygonization

A novel algorithm is presented to create controlled polygonal approximations of implicit surfaces. Using a surface tracking approach, a better approximation is achieved using regions of high curvature rather than spatial subdivision approaches. This algorithm is able to generate an adaptive mesh on the fly, in one step without requiring a post processing re-meshing step. Larger triangles are created in regions of low curvature and smaller ones where curvature is high. This robust method overcomes the sensitivities of surface tracking approaches: it avoids mesh overlap and has support where the gradient is not defined. The polygonization algorithm is guaranteed to be finite in a given volume. Adaptive polygonization is achieved using edge length constraining heuristics calculated during the mesh expansion. The heuristics are based on the curvature information calculated from derivative analysis of the implicit function.

4.1. Polygonization Cycle

Polygonization starts with a seed point computed from a bounding box of the implicit surface volume. The seed point is projected onto the surface using Newton Raphson steps based on gradient evaluation of the field function. The set of points is expanded using newly created vertices. New vertices are created on the tangent plane of existing points and corrected using on principal directions of curvature, expansion angle and point adjacency information.

```
1:  procedure POLYGONIZATION(implicit-surface)
2:      pt0 ← GET-SEED-POINT(implicit-surface)
3:      L ← CREATE-EMPTY-UNEXPANDED-POINT-
    LIST()
4:      M ← CREATE-EMPTY-MESH()
5:      INITIAL-EXPANSION(pt0, L, M)
6:      while IS-NOT-EMPTY(L) do
7:          pt ← GET-POINT-MINIMAL-ANGLE(L)
8:          REMOVE-POINT-FROM-LIST(pt,L)
9:          ptI ←CHECK-COLLISION(pt,L,M)
10:         if ptI = NIL then          ▷ /* no collision */
11:             EXPAND(pt, L, M)
12:         else
13:             LINK − POINTS(pt, ptI)
14:                                    ▷ /* duplicates pt and ptI */
15:             EXPAND − DUPLICATE(pt, L, M)
16:             EXPAND − DUPLICATE(ptI, L, M)
17:         end if
18:     end while
19:     return M
20: end procedure
```

Figure 3: *Adaptive Polygonization Algorithm pseudo-code*

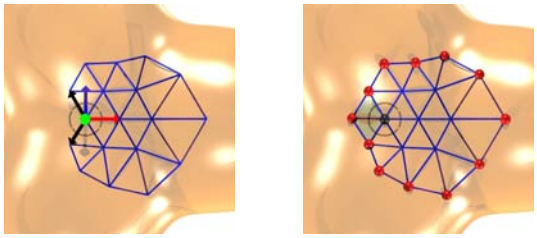

Figure 4: *Expansion of the candidate point using the angle formed with both neighbors, resulting in two new triangles and one new point to the unexpanded point list.*

Triangles are created using new points extruded from an existing surface vertex. Each new point is evaluated until there are no uncalculated points i.e. the surface approximation is complete. The first point expansion generates six new points located on its tangent plane, and six triangles forming a hexagon. Each new point is projected on the surface and its implicit attributes are stored in a list of *unexpanded points*. The algorithm then continues the polygonization cycle selecting unexpanded points to be processed.

Points are selected based on the minimal expansion angle which is defined by the "edge" between a point and two of its neighbours.

4.2. Adaptive Point Expansion

The number of points required to cover an area is defined by the angle θ between the original point and two neighbours that are projected onto the local tangent plane, defined by the principal curvature direction of the original (unexpanded) point. As depicted in Figure 4, the number n of triangles needed to complete the expansion is computed by dividing the angle θ by $\frac{\pi}{3}$ which produces quasi-equilateral triangles. According to the number $(n-1)$ of new edges required, new points are placed on the tangent plane at a distance from the original point given by the heuristic value. The same approach was used by both Hartman [E.98] and Araujo [AJ05a] reducing the possibility of mesh overlap compared to non ordered FIFO or LIFO list policy. Explicit representation of the mesh boundary (such as the front representation used by these approaches does not need to managed).

Before proceeding with the expansion, the distances between the current and the possible new points are estimated. This desired distance can be a constant to produce a uniform polygonization, or based on a heuristic computed using curvature. A collision test searches all the neighbouring points using the heuristic distance value. If no collision is detected, the expansion proceeds and generates new unexpanded points that are projected on the surface using a few iterations of Newton-Raphson steps to achieve predefined precision. Finally, the mesh is updated with new triangles that join new unexpanded points. If the test fails, links between the current point and the nearest point are created and points are duplicated then expanded in order to connect both parts of the mesh. The original point is then removed from the expansion list and the adjacency information of both neighbours of the point are updated considering the new triangulation layout, as shown in Figure 4. This process is repeated until no more expansion is possible. Figure 3 presents the pseudo code of the algorithm.

The final edge length uses a weighted sum based on the candidate point value, its neighbours and the distance between them. This algorithm is therefore less sensitive to high curvature variation and provides a mesh with smooth triangle size transition. Various strategies were followed to produce and control the quality of the adaptive mesh, specifically triangle edge length for the heuristic distance value. Principal curvature, mean, gauss and absolute maximum curvature or shape index are computed. The heuristic-based edge length uses a user-defined scale of the curvature measurement. Thresholds are used to bound the minimum and maximum accepted edge lengths. Binding this value avoids too small or large triangles and the quality of the approximation. This value is used to estimate a local radius of curvature heuristic because derivative analysis of the curvature is not performed. This algorithm also evaluates the heuristic on the neighbours of the candidate point providing a more robust edge length definition related to possible curvature variation.

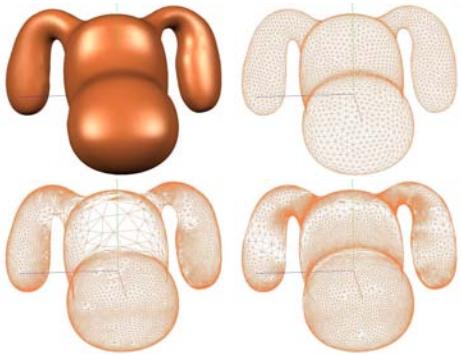

Figure 5: *Constant and Adaptive Polygonization using our algorithm on the ShapShop BlobTree (from top left to bottom right): shaded model, constant heuristic, mean heuristic, gauss heuristic*

4.3. Mesh Overlap Avoidance

Mesh overlap is a typical problem of surface tracking approaches and its avoidance is usually time consuming. The test needs to be efficient and robust considering a dynamic point set. Existing approaches perform the collision test between the boundary of the generated mesh and unexpanded points. The mesh expansion can, however, lead to collisions with interior parts of the mesh. Collision detection, in this research, relies on an octree data structure which is updated dynamically. This structure allows us to spatially store all the points and reduce the neighbour query time. Octree cells are subdivided when they contain more than a predefined number of points. The maximum capacity in our implementation is 15. By doing so, the query which traverses the octree to identify possible colliding points is efficient. Regarding the point expansion, an additional cache is used storing the last visited cells taking advantage of the spacial coherence of the mesh growing process. Two different collision scenarios are considered. If the nearest point is located in the boundary of the mesh, both points are connected and we proceed with a local expansion. If the nearest point is part of the mesh interior, the expansion of the point is simply aborted. These scenarios may arise from unstable values of the implicit function during the polygonization process.

5. Results

Results show that style assignment using an internal node to a hierarchical tree-based system and curvature based polygonization achieves results that are not available with other ISM systems. Complexity and flexibility is achieved by using interactively placed SUNs that allow many different combinations of styles in one image. SUNs can be used to draw the attention to particular parts of a model or image whilst also allowing for a blending region to achieve a more visually smooth result. This complexity is achieved

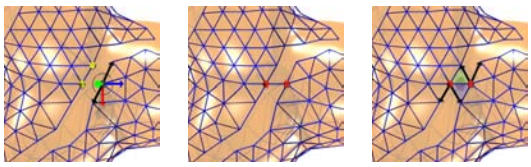

Figure 6: *Mesh overlapping avoidance: the collision test detect colliding points, then a new link is created between the candidate point and the closest colliding point, finally two local expansions are applied avoiding point duplication on the unexpanded point list*

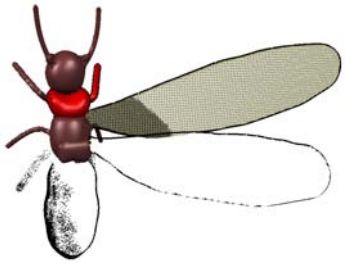

Figure 7: *The termite rendered with a mixture of styles: stipples, sillhouettes, wireframe and filled polygons.*

using: layering and blending of styles; the method of associating styles with subtrees whilst using blend regions to allow smooth transitions between neighbouring styles; and the choice of inheritance properties of family traits .

The main drawback of the method is that some recomputation of field values is necessary, therefore there is a performance cost. This cost is minimised by using hierarchical spatial caching [SWG05] and also allowing SUNs to cache relevant field values.

The main bottleneck is, as is common with implicit surfaces, the field evaluations. For this reason models are generally constructed in relatively low resolution, using a rough to detail approach. As the model is constructed a better rendering of the model is achieved by increasing the resolution (and therefore the approximation to the implicit surface). Finally using the curvature based polygonisation method achieves the best quality rendering in terms of a more accurate approximation to the implicit surface.

The current continuation algorithm for polygonization is fast and the results are adequate for the initial stages of model creation and style assignment. Creating a higher quality image, however, is achieved using a curvature based polygonization.

Figure 8 illustrates the gecko polygonized using the mean curvature heuristic. The modelling tree of the gecko is composed of 13 geometric nodes and 5 style unary nodes. For this example there is a difference regarding the tracking sur-

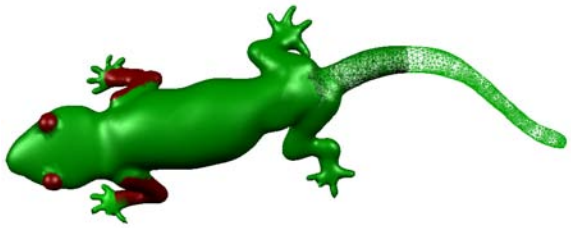

Figure 8: *The gecko rendered using Gaussian curvarure based heuristic.*

Gecko			
	contin	const	gauss
vertices	6782	6597	11261
triangles	13568	13187	22324
time	4.7	6.08	11.01
expansion per s	-	1138.7	1085.9

Table 1: *Statistics for polygonizing the gecko Figs. 8 and 1 Bottom.*

face based polygonization time. The constant heuristic based polygonization (Fig 1) is faster than the gaussian (Fig 8) as less triangles were generated. However, the curvature based polygonization is faster regarding the number of expansions performed per second as less convergence steps are needed to project points on the surface.

6. Conclusions and Future Work

In the research presented in this paper we have described an extension to a hierarchical tree-based ISM system that includes interactively adding style and appearance properties and creating a more accurate curvature based polygonal approximation.

The results presented here illustrate some of the capabilities with using an internal BlobTree node for defining appearances. This includes using field information to evaluate blending regions that extend style choices beyond traditional primitive boundaries.

This method is extendible to other volumetric rendering systems that use scan data. Style nodes can be adapted to use data which is segmented in much the same way as SUN currently are associated with primitives or subtrees.

Unfortunately adding the style nodes has an overhead and requires a traversal of the BlobTree. Future work will redesign the spatial caching to overcome this limitation.

Regarding the underlying polygonal representation of the implicit surface, our approach presents a novel adaptive polygonization algorithm based on curvature information extracted from the scalar field. We follow a surface tracking approach to generate a curvature dependent approximation on the fly. Thanks to several speedup techniques, we achieve a polygonization as fast as the traditional Marching Tetrahedra creating a better approximation.

Acknowledgements

Bruno Araujo was supported by the Portuguese Foundation for Science and Technology, grant reference SFRH/BD/31020/2006.

References

[AGCA06] ALLEGRE R., GALIN E., CHAINE R., AKKOUCHE S.: The hybridtree: Mixing skeletal implicit surfaces, triangle meshes, and point sets in a free-form modeling system. *Graphical Models 68*, 1 (2006), 42–64.

[AHASJI96] A. HILTON A. STODDART J. I. W. T.: Marching triangles: range image fusion for complex object modeling. In *IP'96: Image Processing* (1996), pp. 381–384.

[AJ03] ARAUJO B. D., JORGE J.: Blobmaker: Free-form modelling with variational implicit surfaces. In *Encontro Portugu de Computa Grafica (EPCG)* (2003), pp. 17–26.

[AJ05a] ARAÚJO B., JORGE J. A. P.: Curvature dependent polygonization of implicit surfaces, 2005.

[AJ05b] ARAUJO B. R. D., JORGE J. A. P.: A calligraphic interface for interactive free-form modeling with large datasets. In *SIBGRAPI '05* (2005), p. 333.

[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., 1997.

[BG07] BRUCKNER S., GRÖLLER M. E.: Style transfer functions for illustrative volume rendering. *Computer Graphics Forum 26*, 3 (2007), 715–724.

[BH98] BREMER D., HUGHES J.: Rapid approximate silhouette rendering of implicit surfaces. In *IS'98: Implicit Surfaces* (1998), pp. 155–164.

[BN07] BOUTHORS A., NESME M.: Twinned meshes for dynamic triangulation of implicit surfaces. In *GI '07: Graphics Interface* (2007), pp. 3–9.

[BPK98] BELYAEV A. G., PASKO A. A., KUNII T. L.: Ridges and ravines on implicit surfaces. In *CGI '98: Computer Graphics International* (1998), p. 530.

[BSP*94] BIER E. A., STONE M. C., PIER K., FISHKIN K., BAUDEL T., CONWAY M., BUXTON W., DEROSE T.: Toolglass and magic lenses: the see-through interface. In *CHI '94: Computer Human Interaction* (1994), pp. 445–446.

[BVG05] BRUCKNER S., VIOLA I., GRÖLLER M. E.: Volumeshop: interactive direct volume illustration. In *ACM SIGGRAPH '05 Sketches* (2005), p. 60.

[CDF*06] COLE F., DECARLO D., FINKELSTEIN A., KIN K., MORLEY K., SANTELLA A.: Directing gaze in 3D models with stylized focus. *Eurographics Symposium on Rendering* (2006), 377–387.

[CS02] CERMAK M., SKALA V.: Polygonization by the edge spinning *, 2002.

[CS04] CERMAK M., SKALA V.: Adaptive edge spinning algorithm for poligonization of implicit surfaces. In *CGI '04: Computer Graphics International* (2004), pp. 36–43.

[E.98] E. H.: A marching method for the triangulation of surfaces. *The Visual Computer 14*, 2 (1998), 95–108.

[ER00] EBERT D., RHEINGANS P.: Volume illustration: Non-photorealistic rendering of volume models. In *VIS 2000: IEEE Visualization Conference* (2000).

[FJW*05] FOSTER K., JEPP P., WYVILL B., SOUSA M., GALBRAITH C., JORGE J.: Pen-and-ink for blobtree implicit models. *Computer Graphics Forum (EG '05) 24*, 3 (2005), 267–276.

[GA01] GALIN E., AKKOUCHE S.: Adaptive implicit surface polygonization using marching triangles. In *Computer Graphics Forum* (2001), vol. 20, pp. 67–80.

[HMiBG01] HAUSER H., MROZ L., ITALO BISCHI G., GRER M. E.: Two-level volume rendering. *IEEE Transactions on Visualization and Computer Graphics 7* (2001), 242–252.

[IMT07] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: a sketching interface for 3d freeform design. In *ACM SIGGRAPH '07 courses* (2007), p. 21.

[KHR02] KARPENKO O., HUGHES J. F., RASKAR R.: Free-from sketching with variational implicit surfaces. *Computer Graphics Forum 21*, 3 (2002), 585–594.

[KS01] KARKANIS T., STEWART A. J.: Curvature-dependent triangulation of implicit surfaces. *IEEE Computer Graphics Applications 21*, 2 (2001), 60–69.

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Computer Graphics 21*, 4 (1987), 163–169.

[OBA*03] OHTAKE Y., BELYAEV A., ALEXA M., TURK G., SEIDEL H.-P.: Multi-level partition of unity implicits. *ACM Transactions on Graphics 22*, 3 (2003), 463–470.

[OBS04] OHTAKE Y., BELYAEV A., SEIDEL H.-P.: Ridge-valley lines on meshes via implicit surface fitting. In *ACM SIGGRAPH '04 Papers* (2004), pp. 609–612.

[PASS95] PASKO A., ADZHIEV V., SOURIN A., SAVCHENKO V.: Function representation in geometric modeling: Concepts, implementation and applications a.pasko 1, v.adzhiev 2, a.sourin 3, v.savchenko. In *The Visual Computer* (1995), vol. 11, pp. 429–446.

[PC03] P. CIGNONI C. MONTANI R. S.: Magicsphere: an insight tool for 3d data visualization. *Computer Graphics Forum 13*, 3 (2003), 317–328.

[PLLdF06] PAIVA A., LOPES H., LEWINER T., DE FIGUEIREDO L. H.: Robust adaptive meshes for implicit surfaces. In *SIBGRAPI '06* (2006), pp. 205–212.

[RH04] ROPINSKI T., HINRICHS K. H. (Eds.):. *Real-Time Rendering of 3D Magic Lenses having arbitrary convex Shapes* (2004), Winter School on Computer Graphics.

[SGW06] SCHMIDT R., GRIMM C., WYVILL B.: Interactive decal compositing with discrete exponential maps. In *ACM SIGGRAPH '06 Papers* (2006), pp. 605–613.

[SIW06] SCHMIDT R., ISENBERG T., WYVILL B.: Interactive pen-and-ink rendering for implicit surfaces. In *ACM SIGGRAPH '06 Sketches* (2006), p. 98.

[SWG05] SCHMIDT R., WYVILL B., GALIN E.: Interactive implicit modeling with hierarchical spatial caching. In *Shape Modeling and Applications* (2005), pp. 104–113.

[SWSJ06] SCHMIDT R., WYVILL B., SOUSA M. C., JORGE J. A.: Shapeshop: sketch-based solid modeling with blobtrees. In *ACM SIGGRAPH '06 Courses* (2006), p. 14.

[TZF04] TAI C.-L., ZHANG H., FONG J. C.-K.: Prototype Modeling from Sketched Silhouettes based on Convolution Surfaces. *Computer Graphics Forum 23*, 1 (2004), 71–83.

[VCWP96] VIEGA J., CONWAY M. J., WILLIAMS G., PAUSCH R.: 3d magic lenses. In *UIST '96* (1996), pp. 51–58.

[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.

[WH94] WITKIN A., HECKBERT P.: Using particles to sample and control implicit surfaces. *ACM SIGGRAPH '94* (1994), 269–277.

[WMW86] WYVILL G., MCPHEETERS C., WYVILL B.: Data structures for soft objects. *The Visual Computer 2*, 4 (1986), 227–234.