Complex Skeletal Implicit Surfaces with Levels of Detail

Aurélien Barbier Eric Galin Samir Akkouche

L.I.R.I.S : Lyon Research Center for Images and Intelligent Information Systems
Bâtiment Nautibus, 8 boulevard Niels Bohr
69622 Villeurbanne Cedex, FRANCE

aurelien.barbier@liris.cnrs.fr eric.galin@liris.cnrs.fr samir.akkouche@liris.cnrs.fr

ABSTRACT

Recent research has demonstrated the effectiveness of complex skeletal primitives such as subdivision curves and surfaces in implicit surface modeling. This paper presents a hierarchichal modeling system with an automatic levels of detail management for a simpler modeling with an accelerated rendering. We manage levels of detail with smooth transitions and tree optimizations speeding up visualization by an order of magnitude, which allows an interactive editing of the shapes.

Keywords

modeling, implicit surfaces, complex skeletal primitives, levels of detail, accelerated rendering.

1 INTRODUCTION

A perenial challenge in implicit surface modeling is to provide the designer with high level modeling tools and primitives while maintaining an interactive visualization. The objective that we have already continued for a few years is to develop a coherent system of modeling for the creation of complex virtual environments. These environments are composed of manufactured objects, natural objects and animated characters. Our system is developed around a skeletal implicit surface model named BlobTree [WGG99] that we enrich with complex skeletal primitives. The strategy that we adopted consists in using traditional models as skeletons of implicit primitives to benefit from the intuitive and foreseeable aspect. The field of potential created around these skeletons enables us to control the operations like blending to increase the range of the possible shapes.

Past and recent research has demonstrated the effectiveness of complex skeletal primitives such as curves or surfaces [BS91, CH01]. Moreover, complex surface or volumetric elements may appear in specific applications such as metamorphosis [GLA00, BGA03]. In practice, complex skeletons are seldom used because the implicit surface models suffer from a lack of efficient methods for an interactive or a real-time rendering. Both ray tracing and polygonization techniques require hundreds of thousands of potential field function evaluations to generate an image or create a triangle mesh representation. Unfortunately, potential field evaluations are the more computationally demanding as the skeletons are the more complex. Generally, a significant number of simple primitives is used but it so becomes difficult to control a model well. Thus, there is a need to propose some intuitive complex skeletal primitives combined with accelerated rendering techniques to simply create and edit models like the statue of the figure 1.

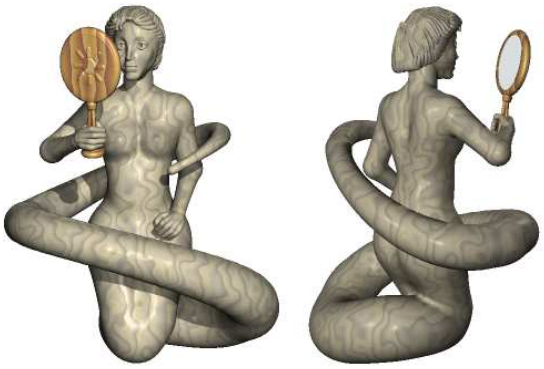

Figure 1: **A complex stone statue defined with LOD - generalized cylinders skeletal primitives.**

We present a framework for modeling complex implicit surface models with levels of detail primitives. [CH01] pioneered the use of skeletal primitives with levels of detail by adapting the subdivision level of some curves and surfaces according to the vizualization requirements. The model used relies on the convolution surfaces which are computationnally expensive.

Our approach provides a more general framework and has

Journal of WSCG, Vol.12, No.1-3., ISSN 1213-6972
WSCG'2004, February 2-6, 2004, Plzen, Czech Republic.

been implemented as an extension of the BlobTree modeling system [WGG99]. The BlobTree organizes simple skeletal primitives in a scene graph. We propose to manage high level primitives with levels of detail by creating instances that are defined as a set of simple atomic primitives organized in a tree structure. The computation of the distance to the skeleton is optimized for each atomic primitive so as to ensure an efficient visualization. Moreover, the number and the type of the generated atomic primitives depend on the required level of detail for the visualization. We also incorporate levels of detail nodes in the scene graph to create smooth transitions between different resolutions. Levels of detail enables us to simplify the model on the fly as needed to accelerate the visualization step.

Traditionnally, implicit surface vizualisation's cost has been lowered by the application of spatial subdivision techniques, as presented in [FGW01], which are memory demanding. Our approach consists in preserving the tree representation of the model by reorganizing the nodes and balancing the tree structure to optimize the bounding boxes hierarchy. This process is applied on the fly during the instanciation process. Moreover, every atomic primitive is highly optimized so that potential field evaluations should be performed efficiently by taking the spatial coherence of queries into account.

Overview

In this paper, we show that complex skeletal primitives don't involve a prohibitive computational cost and allow an intuitive and fast modeling. We present how high level primitives are managed efficiently in our system of animation of characters thanks to levels of detail and tree optimization algorithms.

Section 2 presents the BlobTree model and useful notations. In section 3, we propose a method to define efficient high level primitives with intrinsic levels of detail. Then, we propose a framework for incorporating levels of detail nodes in our tree model which create smooth transitions between the different resolutions of a shape (Section 4). We finally detail some techniques of tree optimization to reduce the complexity of the evaluation of the BlobTree, which enables us to speed-up both ray-tracing and polygonization (Section 5).

2 BACKGROUND AND NOTATIONS

An implicit surface [BBBR97] is mathematically defined as the points in space that satisfy the equation $\mathcal{S} = \{\mathbf{p} \in \mathbb{R}^3, f(\mathbf{p}) - T = 0\}$ where $f(\mathbf{p})$ denotes a scalar field function in space and T a threshold value. Although full blending of simple primitives (points, line segments) has been successfully used to create clouds or viscous liquids [DC95], experiments demonstrated that this model is inadequate for representing complex models. Since, [PASS95] and [WGG99] have proposed two similar hierarchical models that incorporate boolean operations, blending and deformation in a unified system.

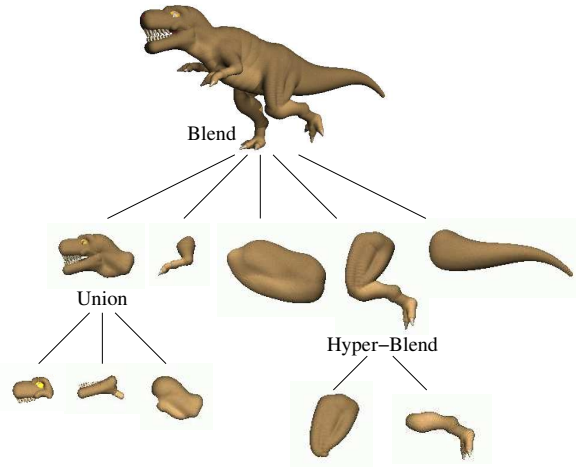

Figure 2: **A simplified representation of the BlobTree structure of a tyrannosaurus-rex model.**

The BlobTree model [WGG99] is characterized by a hierarchical combination of skeletal primitives organized in a tree data-structure (Figure 2). The nodes of the tree hold boolean, blending and warping operators. Skeletal primitives generate potential field contributions, denoted as f_i, that are decreasing functions of the distance to a skeleton: $f_i = g_i \circ d_i$ where $g_i : \mathbb{R}_+ \to \mathbb{R}$ is the potential field function, and $d_i : \mathbb{R}^3 \to \mathbb{R}_+$ refers to the distance to the skeleton. Both the skeleton and the distance function d_i characterize the shape of the element, whereas the potential field function g_i defines the way elements blend together.

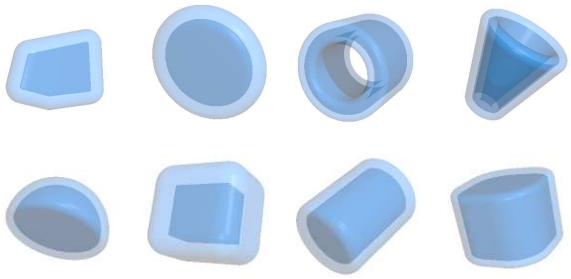

Figure 3: **Atomic skeletal primitives with an opaque skeleton and the corresponding transparent potential field.**

Our system implements a large set of primitives of any dimension (Figure 3). Circle, circular arc, quadric and cubic polynomial curves extend the familly of curve skeletons. Surface skeletons include triangle, rectangle, convex polygon, disc, hollow cylinders and cones. We have implemented primitives with surface of revolution skeletons defined by sweeping a quadric and cubic curve around an axis. Voluminal skeletons include sphere, hemisphere, cone, box, cylinder, and cylinder-box. Finally, our system incorporates super-ellipsoids

[Bar81] which are based on $\mathcal{L}^p$ metrics, ratio-quadrics [BS96] whose field function is more efficient to compute, and rotational and translational primitives described in [CBS96].

All those skeletal primitives implement a specific and optimized distance function $d_i(p)$. Therefore, they will be refered to as atomic primitives. More complex primitives with levels of detail will be instanciated as a combination of those atomic primitives.

3 HIGH LEVEL PRIMITIVES WITH LEVELS OF DETAIL

The very issue consist in defining tools offering to the designer complex primitives while guaranteeing efficient computing times for an interactive editing. We present some high level primitives controlled by a small set of parameters that automatically manage levels of detail and generate sub-trees of atomic primitives. Indeed, we split complex skeletons into a union of atomic curve, surface and voluminal skeletons that may be processed efficiently to ensure an optimized visualization. Using the union opertor guarantees that no bulging effect appears. The subdivision level used to create the BlobTree representation of the high level primitives is defined as a function of the input level of detail. The scalar value $\lambda \in [0,1]$ denotes the level from coarsest to finest. The BlobTree generation process adapts the generated skeletal atomic primitives according to the variations of the thickness for a given level of detail so as to generate atomic primitives with faster distance computation.

The concept of high level primitives is general within the framework of our model. Therefore, many classes of complex primitives may be added. Our current implementation includes three primitives whose skeletons are generalized cylinders, surface patches with varying thickness, and volumes of revolution.

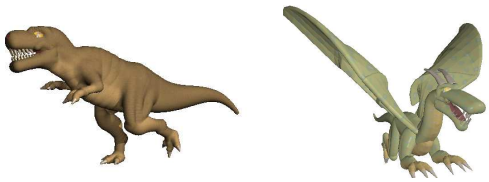

Figure 4: **A tyrannosaurus-rex and a dragon defined with LOD - high level primitives.**

Curve-based LOD primitives

Generalized cylinders are often used to design tubular organic shapes of varying radius. We created the tyrannosaurus-rex (Figure 4 left), the statue (Figure 1) and the body of the dragon (Figure 4 right) very efficiently and intuitively by blending a few generalized cylinder skeletons. A generalized cylinder skeleton is parameterized by a sweeping curve, denoted as $\mathbf{c}(t)$ and a freely varying radius function, denoted as $r(t)$ along the support curve (Figure 5). The system discretizes the curve into a set of n line segments whose vertices are denoted as $\mathbf{v}_i$, $i \in [0, n-1]$, depending on the level of detail coefficient $\lambda \in [0,1]$. For each vertex $\mathbf{v}_i$, the system computes the corresponding radius parameter r_i and creates a sphere primitive while for each line segment a cone skeleton is created. The final potential field is defined as the union of the potential fields of the generated primitives.

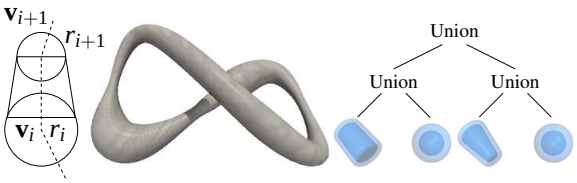

Figure 5: **A LOD - generalized cylinder primitive and its corresponding BlobTree representation.**

Several optimizations have been included in the primitive generation process. If the end radii of a piece are constant, the system instanciates a cylinder atomic skeletal primitive that is faster to process than a cone. In the same way, if the end radii are null, a line segment skeleton is generated. Eventually, if the required level of detail is not very high, a cone with close end-radii values is approximated by a cylinder whose potential field function is faster to evaluate. Moreover, when creating the BlobTree representation, our evaluation engine organizes primitives into an optimized hierarchy of union nodes (Figure 5). This approach takes advantage of spatial coherence and reduce computations when performing potential field computation queries as described later in Section 5.

The snake-woman statue (Figure 1) involves 257 LOD generalized cylinders but 176 of them were used to create the hair style. The final model generated for rendering at the maximum level of detail features 8600 optimized atomic primitives. It would be difficult to create such a complex model by editing each single primitive.

Surface-based LOD primitives

[AC02] recently proposed to use subdivision surfaces as skeletons in the context of convolution surfaces by adaptively meshing the surface into a set of triangles. We have adapted different types of surfaces to our system. Subdivision surfaces as well as parametric surfaces generate a set of triangles that approximate the surface at a given level of detail. Triangles are used to create skeletal atomic primitives that are unioned in a sub-tree to create the final approximation of the distance surface.

We have added some extra control by defining a parameterized function that represents the thickness of the skeleton at each point of the surface which enables us to create more complex shapes. Figure 6 shows the BlobTree representation of this high level primitive. The system creates a triangle mesh of the surface depending on the

level of detail coefficient $\lambda \in [0,1]$. Then, for each vertex $\mathbf{v}_{ij}$ the system computes the thickness parameter t_{ij} and a vertex normal $\mathbf{n}_{ij}$ by averaging the normals of the neighbouring triangles. A prism polyhedral primitive whose six vertices are computed by offsetting the three vertices of the triangle in the direction $\pm t_{ij}\,\mathbf{n}_{ij}$ is generated for each triangle and the borders are processed as described for generalized cylinders. The final potential field is defined as the union of the potential fields of the generated primitives.

Figure 6: **A surface patch with varying thickness primitives, and its corresponding BlobTree representation.**

Prisms are computationally expensive skeletons. As for generalized cylinders, several optimizations were included in this otherwise simple process. If the thickness radii at the vertices of the base triangle are the same, then the prism has parallel faces and the general prism skeletal element is replaced by an optimized version. Eventually, the prism primitives simplify to triangles if the thickness function is null.

Let us inspect the creation of the dragon model (Figure 4). The body was created by blending 79 generalized cylinder skeletons. The end part of the wing has a small thickness, whereas the section near the junction with the body is thicker. Each wing was designed using surface patches with variable thickness skeletons. Each patch was controlled by four cubic spline curves and the varying thickness was controlled by four cubic spline functions that represent the thickness along the splines of the surface patch. The final model used for the rendering at the maximum level of detail contains 5337 optimized atomic primitives.

Volumes of Revolution with LOD

It could be possible to use the preceding high level primitive to create volumes of revolution models but we propose an optimized method for these shapes. In the general case, volumes of revolution are generated by sweeping a two dimensional closed surface around an axis. Here, we restrict to surface defined as two dimensional generalized cylinders. The bottle and glass models in Figure 8 were created using this technique. The BlobTree generation process is similar to generalized cylinders.

First, the system discretizes the profile curve into a set of n line segments whose vertices are denoted as $\mathbf{v}_i$, $i \in [0, n-1]$. For every vertex $\mathbf{v}_i$, the system computes the curve normal denoted as $\mathbf{n}_i$ and evaluates the corresponding thickness parameter t_i (Figure 7).

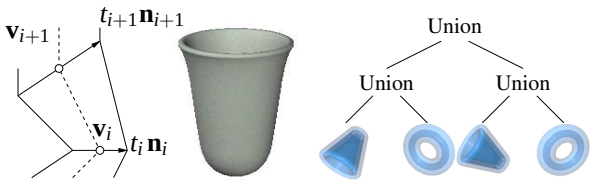

Figure 7: **A volume of revolution and its corresponding BlobTree representation.**

Then, for every line segment, the vertices of the four sided polygon $\mathbf{v}_i \pm t_i\,\mathbf{n}_i$ and $\mathbf{v}_{i+1} \pm t_{i+1}\,\mathbf{n}_{i+1}$ are computed and a primitive of revolution is created with this polygon as revolution skeleton, which is a pieced hollow cone of varying radii primitive. The system also generates torus primitives with center $\mathbf{v}_i$, major radius $\|\mathbf{n}_i\|$ and minor radius r_i to create smooth junctions between revolution primitives. The final volume of revolution is defined as the union of the generated optimized atomic primitives. Several optimization steps have been incorporated in this process. If the thickness parameter is null, we create hollow cones primitives, hollow cylinders primitives and even surface of revolutions with quadric or cubic polynomial profil curves primitives.

Figure 8: **Bottles and glasses of wine and champagne modeled with LOD - volume of revolution primitives.**

4 MANAGING LEVELS OF DETAIL

One of the weaknesses of implicit surface models compared to triangle meshes [MH99] has long been the lack of methods for generating models at different levels of detail. Since implicit surfaces are generated by a field function in space, defining levels of detail is equivalent to generating field functions at different resolutions. In our model, levels of detail may be handled by high level primitives as described above, or by specific nodes in the BlobTree itself. In this section, we propose a coherent framework for managing levels of detail with smooth transitions between the representations of an implicit model.

Multi Representation Nodes

In the simplest type of levels of detail, the different representations are models of the same object containing different primitives. Thus, our system implements a new levels of detail node, denoted as $\mathcal{L}$, in the BlobTree's definition. $\mathcal{L}$ is an n-ary node that selects one of its children to represent the field function defining an object for a given input level of detail (Figure 9).

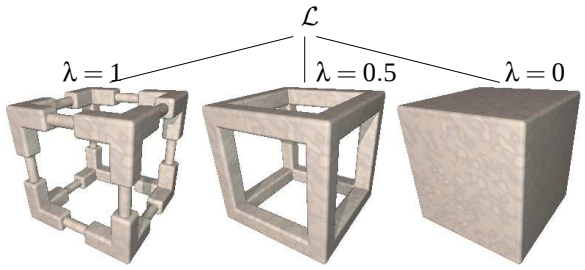

Figure 9: **Multi-representation node describing a cubic structure.**

As for meshes, one problem is that popping may occur when switching between different levels of detail. Interpolating two models, *i.e* summing the weighted contribution of the two field functions at different resolution may be used to avoid this problem. However, this process requires that two field functions should be evaluated which slows down ray-tracing and polygonization significantly. Therefore, we have developed geomorph levels of detail models in our system that are faster to evaluate.

Geomorph Levels of Detail

Geomorph levels of detail were first proposed in [Hop96] as a set of mesh models created by simplification. Instead of using alpha blending to create a smooth transition from one model to the next, the vertices of one model move to the other model's vertex locations. We adapt the concept of geomorph to the BlobTree by progressively simplifying the primitives and the nodes of a complex model. Simplifications may be obtained in two ways, either using automatic techniques, such as skeleton simplification and field function fading, or using parameterized models edited by the designer.

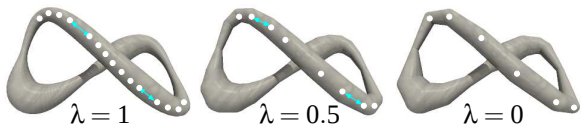

Figure 10: **Smooth transitions between models.**

By using a progressive mesh representation for curve and surface supports of our high level primitives, we can define them as progressive levels of detail primitives (Figure 10). In this context, the creation of a multiresolution model relies on an automatic curve or triangle mesh simplification process as described earlier. Some primitives however include specific simpler representations. For instance, a cone primitive with a small slope automatically transforms to a cylinder as the level of detail decreases.

Simplifications are also obtained by smoothly reducing the influence of the field function of a given primitive or sub-tree to 0. This may be easily achieved by parameterizing the potential functions by a weighting coefficient that decreases as the required level of detail gets smaller.

In the most general case, our system allows the designer to freely parameterize skeletal elements and high level primitives as functions of a given input level of detail variable. Contrary to previous methods, this technique requires some modeling skills, but provides a very tight control over the shape transformation. Figure 11 shows a column at different levels of detail. The left image show the column at maximum resolution. The next three models show progressive simplification, as primitives that add detail are smoothly removed. The rightmost image shows a column where small details disappeared. The top and bottom parts of the column are simple box primitives, and the cylinder needs to grow to fill the gap created when removing the top and bottom discs.

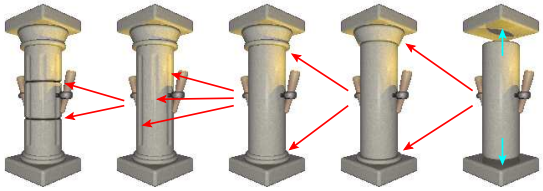

Figure 11: **Column with geomorph levels of detail. See figure 16 for a use of each level of detail depending on the distance to the viewer.**

Figure 12 represents the dragon model at three different levels of detail. The small images show the dragon at the corresponding viewing scales. Figure 13 show the use of each level of detail with a swarm of dragons. Figure 16 shows a temple with some columns forming a path and a statue in the back. All the elements of this scene were created with levels of detail primitives and nodes, which accelerated ray-tracing process by a factor of 3.

Table 1 reports the timings for ray-tracing different models at different levels of detail at 704×396 pixels resolution. Figures show that using models at a low level of detail accelerates visualization by an order of magnitude, without noticeable artifacts.

Objects	Coarse	Medium	Accurate
tyrannosaurus-rex	83	208	639
dragon	94	458	770
statue	92	230	706

Table 1: **Timings (in seconds) for ray-tracing complex models at different levels of detail ($0, 0.5, 1$).**

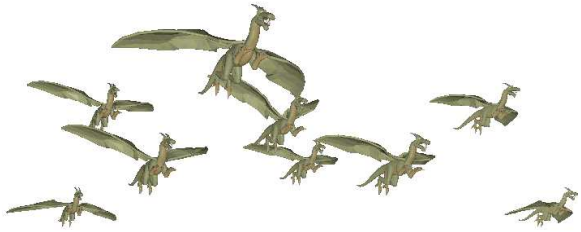

Figure 13: **A swarm of flying dragons.**

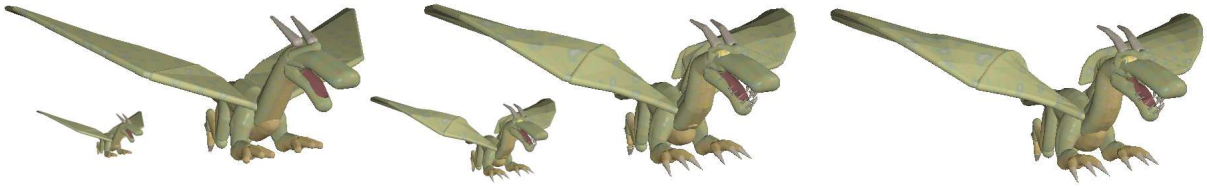

Figure 12: **A dragon at different levels of detail. The wings are modeled with surface patches with varying thickness skeletons and most of the body with generalized cylinder skeletons.**

5 EFFICIENT PROCESSING

The use of the high level primitives allows an effective and intuitive editing. However the resulting BlobTree may involve some redundancies in the nodes which leads to an unbalanced tree. An optimization step is necessary for an efficient processing. We present some classical approaches applied to the BlobTree structure to reduce the computational rendering time. Spatial coherence is used to reduce the potential field computation either for ray-tracing or polygonizing the optimized BlobTree.

Tree Optimization

In the general case, the number of primitives that effectively contribute to the final potential field at a given point in space is small compared to the overall number of primitives involved in a particular model. Traditionally, visualization's computational cost has been lowered by the application of spatial subdivision methods. [FGW01] proposed an algorithm that uses a voxel decomposition of space to split the model into a set of simpler BlobTree models located in the cells of the voxel. However, this approach is only applied as a pre-processing step before visualization and is memory demanding.

We aim at preserving a unique tree representation of the BlobTree. The optimization process has to be able to be invoked at any time, even during the creation of a shape, so as the optimized model may be further edited as needed. Our system automatically rewrites the scene graph of the BlobTree in order to optimize its internal bounding boxes hierarchy.

First, the system removes the nodes of affine transformation by integrating them into the parameters of the skeletal primitives. As prescribed in [FGW01], we use the following result: $Affine(Op\{\mathcal{N}_i\}) = Op\{Affine(\mathcal{N}_i)\}$ where Op is a boolean or a blending operator and $\mathcal{N}_i$ denotes a child node of the considered n-ary operator. Unfortunately, warping operators block this otherwise simple process in the general case. We have optimized the algorithm so that rotations and translations can be cast through some warping nodes such as twist.

The second step merges nodes of the same type to reduce the depth of the BlobTree (Figure 14 (a)). Let $\mathcal{N}$ be a n-ary node of the tree holding boolean or blending operators. If one of its child node $\mathcal{N}_i$ is of the same type, then we move the children nodes of $\mathcal{N}_i$ to $\mathcal{N}$ and remove $\mathcal{N}_i$. This tree merging step is recursively applied throughout

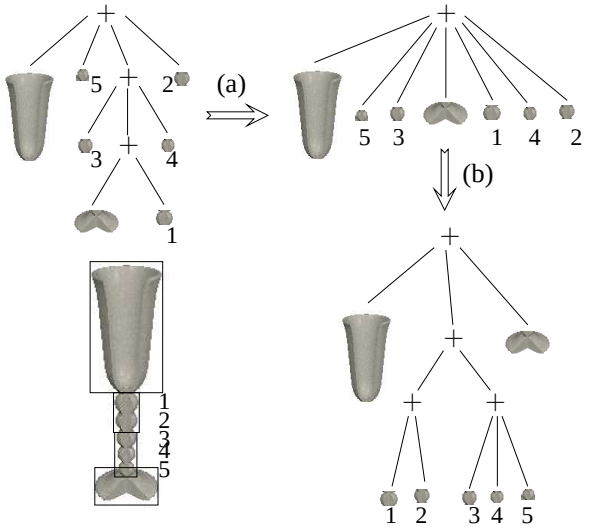

Figure 14: **Optimization of the scene graph: merge the nodes of the same type to reduce the depth and balance the tree to create an optimized bounding boxes hierarchy.**

the BlobTree thanks to the previous algorithm that ensures that affine nodes do not block the process.

For the third and last step (Figure 14 (b)), we rely on the algorithm presented in [GS87] to split the boolean and blending nodes so as to create an optimized hierarchy of bounding boxes of their child nodes.

Spatial Coherence

One of the most expensive step in the evaluation of the field function at a given point in space $\mathbf{p}$ is the computation of the distance $d(\mathbf{p})$ to complex skeletons. When ray-tracing an implicit surface, the ray model intersection [KB89] is found by sampling the potential field along the ray to isolate the roots and converge. The queries could benefit from spatial coherence. The polygonization [Blo88] using a voxel decomposition of space may also be rewritten in order to take advantage of the coherence of samples that are located on the lines of the voxel grid.

Recall that for every ray, we first recursively check if it intersects the bounding box of each node of the BlobTree. If no intersection occurs for a node, the whole sub-tree is disabled so as to avoid the propagation of unnecessary further queries down the sub-tree. We propose to speed-up the future potential field computations of the enabled nodes by pre-computing and caching constant

terms. Skeletal primitives split the ray into different intervals where the distance to the skeleton is unambiguously defined, and compute the closed form equation of the squared distance to their skeleton $d(\mathbf{p}(t))$ whenever possible (Figure 15). We identify two classes that rely on the same pre-processing method explained below: surfaces of revolution and polytopes. Note that some other primitives, such as polynomial curves, may be optimized in several ways by caching constant terms as well.

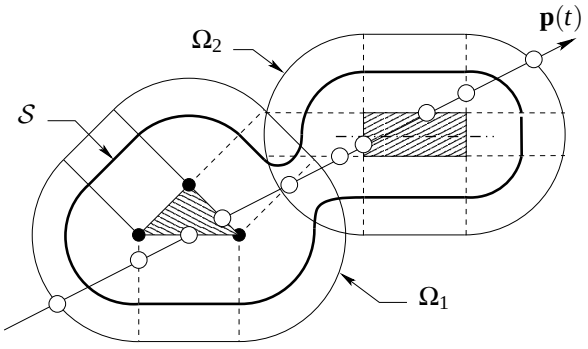

Figure 15: **The ray of the queries is splitted in independant intervals. Ω_1 and Ω_2 are the region of influence of the primitives. $\mathcal{S}$ is the final iso-surface.**

Surfaces of revolution as well as the circle, disc, cylinder (Figure 15 right), cone, and tube primitives may be processed by caching the constant terms of the equation of the projection of the point $\mathbf{p}(t)$ on the axis, denoted as $a(t)$, and by caching those of the equation of the distance to the axis, denoted as $r(t)$. In the general case, the computation of the distance function $d(\mathbf{p}(t))$ relies on the evaluation of the quadric polynomial $r(t)$ and the linear function $a(t)$, which may be performed by a mere 3 adds and 3 multiplies. This is much faster than the direct computation of the projection onto the axis $a(\mathbf{p})$ and the evaluation of the radial distance is $r(\mathbf{p})$ that cost 8 adds and 7 multiplies. Specific surfaces of revolution such as cylinders may be optimized even more.

Whatever the dimension of the polytope, the region of influence Ω_i around the skeleton is a rounded polytope defined by sweeping a sphere over the skeleton. We split Ω_i into non-overlapping sub-regions in which the computation of the distance is straight forward and defined as the distance to a point, a line or a plane (Figure 15 left). This decomposition enables us to compute the closed form expression of the distance $d(\mathbf{p}(t))$ as a piecewise quadric polynomial in t which greatly speeds up computations.

Timings

We have developed an object oriented class hierarchy that implements all the features described through out this paper. Tables 2 and 3 report timings for ray-tracing and polygonizing the models shown in Figures 1, 4, and 8. The first column of Table 2 report timings without any pre-processing step. The second column show accelerations obtained after balancing the BlobTree. The third column reports timings corresponding to a balancing preprocessing step combined with the optimized potential field computation algorithm. Timings were performed on a P-*IV* processor at 1.6 GHz with 256 Mo of memory.

Objects	Standard	Balanced	Optimized
wine bottle	69	67	25
wine glass	56	53	15
champagne bottle	49	48	21
champagne glass	33	30	5
tyrannosaurus-rex	1786	258	133
dragon	2392	399	241
statue	2841	373	59

Table 2: **Ray-tracing - timings in seconds. The technique is based on an optimized version of the original Lipschitz based method first described in [KB89]. Images were generated at 512×512 pixels size.**

Objects	Balanced	Optimized	Triangles
wine bottle	23	17	159492
wine glass	16	10	134448
champagne bottle	33	23	213012
champagne glass	15	10	89168
tyrannosaurus-rex	34	22	89888
dragon	54	40	118140
statue	80	43	298796

Table 3: **Polygonization - timings in seconds. The technique relies on a brute force 256^3 voxel decomposition of space.**

Timings show that balancing the tree structure of the BlobTree plays a major part in the acceleration of the computations for huge models. For instance, the ray-tracing time for rendering the statue model drops from 2841 seconds to 373 seconds when balancing the BlobTree. In contrast, the balancing process produced almost no acceleration for models with few skeletal primitives since they were already almost balanced.

In all cases, the accelerations obtained by preprocessing complex primitives for every ray are significant. Accelerations are the more significant as the skeletal primitives are complex. For instance, the ray-tracing time for rendering the champagne glass model drops from 30 seconds to 5 seconds by caching constant terms in the evaluation of the potential field. In contrast, accelerations are less important for the dragon model: timings report ray-tracing time decreasing from 399 seconds to 241 seconds, as most skeletal atomic primitives are quite simple (triangles, spheres and cones) where accelerations are smaller.

6 CONCLUSION

In this paper, we have proposed high level primitives that automatically manage levels of detail in a natural way. Those tools provide a great modeling flexibility and power to the designer who can easily control the shape. We also have introduced a new levels of detail operator that add multi-representation to the BlobTree model

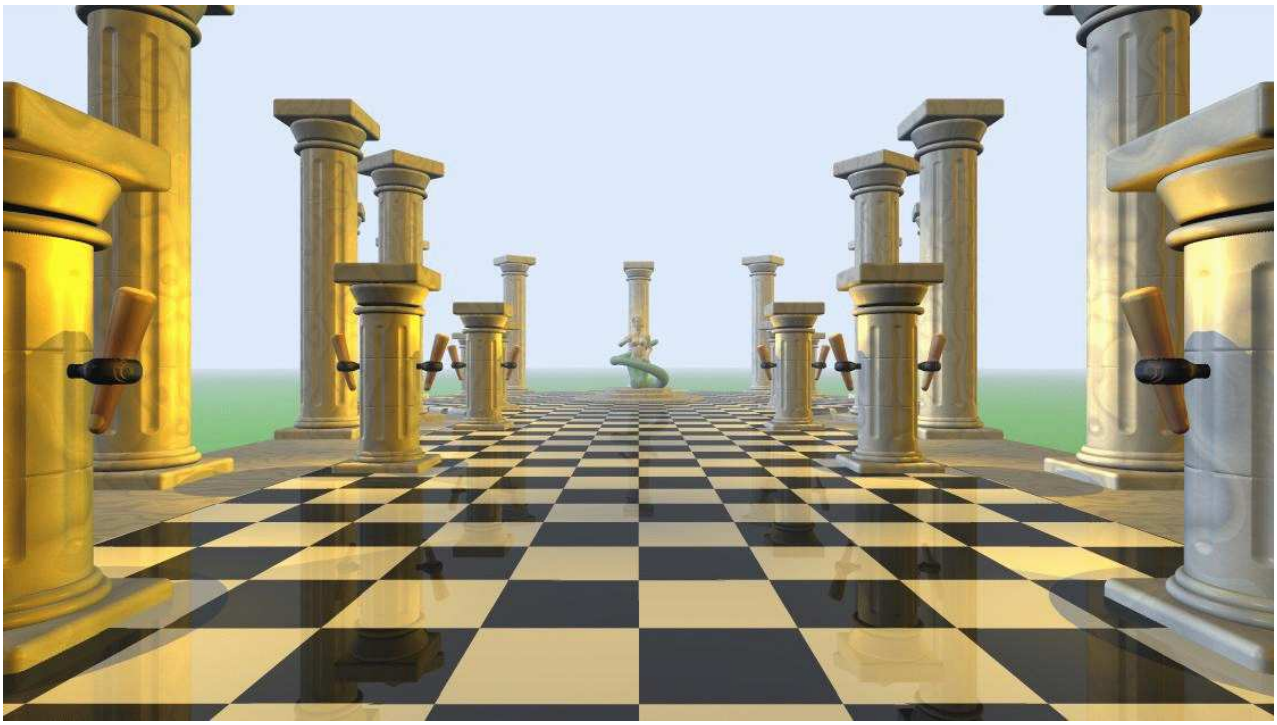

Figure 16: **A temple with columns and statue with levels of details.**

and we have presented a general framework guaranteeing smooth transitions between the different levels of detail. Those modeling tools are used in an interactive environment thanks to several optimized techniques that speed up by an order of magnitude the computation of the potential field for both ray-tracing and polygonization algorithms. The automatic generation of different models used in a multi-representation operator will be studied as a future work. Moreover, even if the techniques of acceleration which we propose are effective, many ways remain to explore for a very fast polygonization.

7 REFERENCES

[AC02] Alexis Angeledis and Marie-Paule Cani. Adaptive implicit modeling using subdivision curves and surfaces as squeletons. In *Proceedings of Solid Modeling, Saarbrucken, Germany*, pages 45–52, 2002.

[Bar81] Alan Barr. Superquadrics and angle-preserving transforms. *Computer Graphics and Applications*, 1(1):11–23, 1981.

[BBBR97] Jules Bloomenthal, Chandrajit Bajaj, Jim Blinn, and Alyn Rockwood. *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers, 1997.

[BGA03] Aurélien Barbier, Eric Galin, and Samir Akkouche. Controlled metamorphosis of animated objects. In *Proceedings of Shape Modelling International, Seoul, Korea*, pages 184–196, 2003.

[Blo88] Jules Bloomenthal. Polygonization of implicit surfaces. In *Computer Aided Geometric Design*, volume 5, pages 341–355, 1988.

[BS91] Jules Bloomenthal and Ken Shoemake. Convolution surfaces. In *Computer Graphics (Siggraph proceedings)*, volume 25, pages 251–256, 1991.

[BS96] Carole Blanc and Christophe Schlick. Ratioquadrics: an alternative method for superquadrics. *The Visual Computer*, 12(8):420–428, 1996.

[CBS96] Benoit Crespin, Carole Blanc, and Christophe Schlick. Implicit sweep objects. *Computer Graphics Forum*, 15(3):165–174, 1996.

[CH01] Marie-Paule Cani and Samuel Hornus. Subdivision-curve primitives: a new solution for interactive implicit modeling. In *Proceedings of Solid Modeling*, pages 82–88, 2001.

[DC95] Mathieu Desbrun and Marie-Paule Cani. Animating soft substances with implicit surfaces. In *Computer Graphics (Siggraph proceedings)*, volume 29, pages 287–290, 1995.

[FGW01] Mark Fox, Callum Galbraith, and Brian Wyvill. Efficient use of the BlobTree for rendering purposes. In *Proceedings of Shape Modelling International, Genova, Italy*, 2001.

[GLA00] Eric Galin, Antoine Leclercq, and Samir Akkouche. Morphing the blobtree. *Computer Graphic Forum*, 19(4):257–270, 2000.

[GS87] Jeffrey Goldsmith and John Salmon. Automatic creation of object hierarchies for ray-tracing. *Computer Graphics and Applications*, 7(5):14–20, 1987.

[Hop96] Hugues Hoppe. Progressive meshes. In *Computer Graphics (Siggraph proceedings)*, pages 99–108, 1996.

[KB89] D. Kalra and Alan Barr. Guaranteed ray intersection with implicit surfaces. In *Computer Graphics (Siggraph proceedings)*, volume 23, pages 297–306, 1989.

[MH99] Thomas Möller and Eric Haines. *Real time rendering*. A.K. Peters, Ltd, 1999.

[PASS95] Alexander Pasko, Valery Adzhiev, Alexei Sourin, and Vladimir Savchenko. Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer*, 11(8):429–446, 1995.

[WGG99] Brian Wyvill, Andy Guy, and Eric Galin. Extending the CSG tree (warping, blending and boolean operations in an implicit surface modeling system). *Computer Graphics Forum*, 18(2):149–158, 1999.