Controlled Metamorphosis of Animated Objects

Aurélien Barbier, Eric Galin, Samir Akkouche

L.I.R.I.S
Université Claude Bernard Lyon 1
69622 Villeurbanne Cedex, FRANCE

{abarbier,egalin,sakkouch}@ligim.univ-lyon1.fr
http://www710.univ-lyon1.fr/~abarbier

Abstract

Although many animation, deformation and metamorphosis techniques have been proposed, the simultaneous combination of those three transformations has never been addressed for three dimensional objects so far. This paper presents a framework for controlling metamorphosis between two animated implicit models built from skeletal elements. It relies on the animated BlobTree model that encompasses the animation and metamorphosis in an unified and coherent fashion. Our method is general and supports a wide range of animation systems such as key-frames or physically based systems.

Keywords: control, metamorphosis, animation, implicit surfaces, special effects

1 Introduction

Animation and metamorphosis have been in the highlights of the computer graphics community to meet the increasing demand of the entertainement industry. A wide range of animation, deformation and metamorphosis techniques have been proposed in the literature for the past few years. Moreover, existing metamorphosis techniques that operate on three dimensional models transform still objects only. This paper adresses the metamorphosis of animated shapes and focuses on the control of the transformation.

One major concern in shape metamorphosis is the preservation of shape coherence during the transformation. Shape coherence becomes even more difficult when transforming animated objects. Animation coherence should be also preserved to create convincing and natural looking transitions between animations. Eventually, the animator should be able to control the shape and the speed of the transformation. or edit the transformation to add local deformations to create special effects.

1.1 Related work

We present our approach in the context of two related areas : metamorphosis and animation.

Existing metamorphosis techniques are strongly dependent on the underlying geometric model. Mesh models, which are prominently used in major modelling and animation software, are difficult to metamorphosize, especially when dealing with surfaces of different topology. In contrast, skeletal implicit surfaces lend themselves for transforming complex shapes of arbitrary topology.

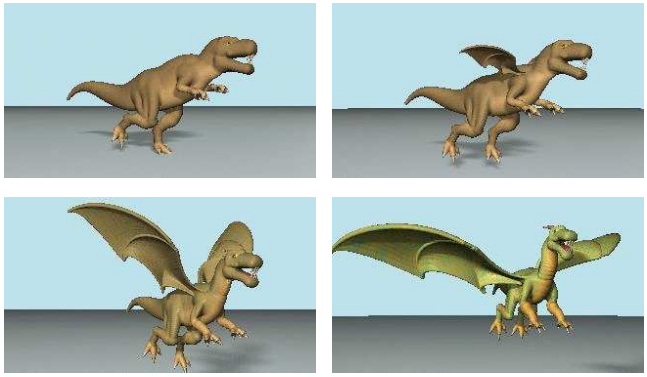

Figure 1: Metamorphosis between a walking Tyrannosaurus-Rex and a flying Dragon Characters. The back legs of the T-Rex progressively stop moving while the wings grow and start flattering as the character takes off.

Metamorphosis Most mesh metamorphosis techniques first create a graph of correspondences matching the vertices of the two models, possibly merging their topologies, and define the transformation by performing an interpolation between the geometries [Lazarus and Verroust 1998; Alexa 2001]. In practice, the correspondence process may be awkward in the general case, and dealing with shapes of different topology remains difficult.

Several metamorphosis techniques for voxel models have been proposed [Hughes 1992; He et al. 1994; Lerios et al. 1995]. One major drawback is that good visual effects cannot be obtained unless using a fine sampling of the objects, and the computation cost becomes the more prohibitive as the size of the sampling grid increases.

Skeletal implicit surfaces have proved to be the most efficient model for metamorphosizing shapes of arbitrary topology [Pasko and Savchenko 1995; Galin and Akkouche 1996; Galin et al. 2000]. Shape coherence, which is essential for creating convincing transformations, may be preserved by controlling the paths of interpolating skeletal elements. Existing techniques only address the transformation of still objects however. Our approach presents a unifying framework that allows the deformation and metamorphosis during animation and provides a high level of control to ensure both shape and animation coherence.

Animation Both key-frame and physically based animation techniques have been applied to animate implicit surfaces. One of the major applications of implicit surfaces in physically based animation has been the simulation of viscous liquids [Desbrun and Cani 1995], soft inelastic models based on particle systems [Miller and Pearce 1989; Tonnesen 1991]. Recently, they have been successfully applied to characterize the smooth surface of fluids such as water [Foster and Fedkiw 2001]. Moreover, implicit surfaces lend themselves for collision detection and precise contact modeling between flexible solids [Gascuel 1993].

Earlier key-frame animation systems produced animation by simply moving the skeletons of the implicit model along a frame curve, or by applying deformations in a warped space [Bloomenthal et al. 1997]. The animation of the Frep model is proposed in [Fausett et al. 2000]. The concept of animation tracks in the BlobTree model was first proposed in [Nur et al. 2001]. Our approach relies on a BlobTree structurewhose nodes are fully parameterized by functions of time, which may be of any type. Thus, our approach can handle both physically based and key-frame based animation systems.

1.2 Overview

The BlobTree [Wyvill et al. 1999] is a powerfull implicit surface modeling system that combines blending, boolean and warping operations performed on skeletal primitives. In [Galin et al. 2000], we have proposed an efficient approach for metamorphosizing two BlobTree models. The method provides the animator with a hierarchical control over the transformation through the use of a graph of correspondences matching elements or whole sub-trees of the arguments models.

In this paper, we present an original technique for metamorphosizing two animated models defined by skeletal implicit surfaces. Specifically, we define the animorphosis as a metamorphosis performed between two animated objects. We show that parameterizing the BlobTree by functions of time is a natural way to create complex animations and transformations. This approach encompasses the animation and metamorphosis in a unified and coherent model.

The evolution of the model is defined by a generic BlobTree structure combined with classes of time varying parameters. Assigning evolution functions to the parameters of every node in the BlobTree leads to an increasing number of time dependent evolution functions for complex models. In practice, the hierarchical structure of the BlobTree enables the animator to control the whole animation or transformation by tuning but a few parameters located at the top nodes in the BlobTree. A very tight control may be achieved if necessary by tuning the parameters of lower nodes, which provides a level of detail in the control of the animation.

Whenever the animation of the source and target models is based on key-frames, we provide a semi-automatic method of animorphosis that creates a generic animated BlobTree controlled by key-frame parameters. This approach takes advantage of the hierarchical structure of the BlobTree to provide an intuitive hierarchical control over the transformation. This technique ensures that both shape and animation coherence are preserved by the use of well chosen time steps.

The remainder of this paper is structured as follows. Section 2 recalls the fundamentals of the BlobTree and details the control parameters that are used to produce animations. Section 3 presents the animorphosis concept, focuses on shape and animation coherence and then presents the semi-automatic key-frame animorphosis method. Finally, section 4 shows the results for the different kinds of animorphosis.

2 Fundamental concepts

In this section, we present the animated BlobTree model. It is designed as an extension of the original static BlobTree model [Wyvill et al. 1999] that only creates still objects. Although our extended model simply relies on the parameterization by functions of time of each parameter of the BlobTree model, we here present it entirely for the sake of clarity.

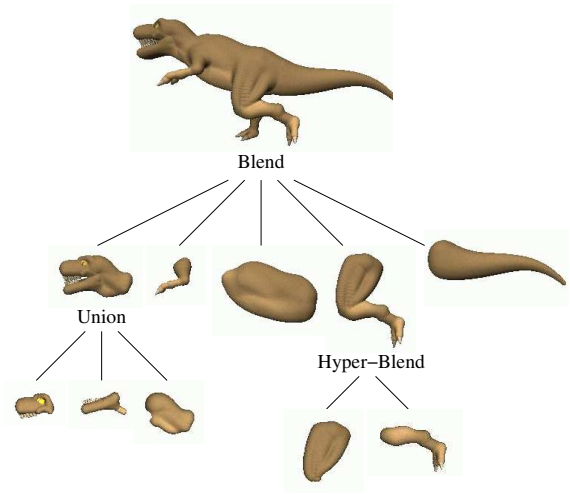

Figure 2: Synthetic representation of the BlobTree structure of a Tyrannosaurus-Rex model.

2.1 The BlobTree model

The BlobTree is characterized by a hierarchical combination of primitives organized in a tree data-structure. The nodes of the tree hold blending, boolean operators and warping operators, whereas the leaves are characterized as skeletal elements. Figure 2 represents a Tyrannosaurus-Rex model created with sphere, cylinder, cone and spline skeletal elements combined with blending and boolean operators.

The field contributions of the skeletal primitives are decreasing functions of the distance to a skeleton $f_i = g_i \circ d_i$ where $g_i : \mathbb{R}_+ \rightarrow \mathbb{R}$ is the potential function, and $d_i : \mathbb{R}^3 \rightarrow \mathbb{R}_+$ refers to the distance to the skeleton. The evaluation of the field function at a given point in space is achieved by recursively traversing the BlobTree, either evaluating the field functions at the leaves of the tree or combining the field function values returned by the children of a given node.

2.2 Animating the BlobTree

The animated BlobTree model is characterized by the hierarchical generic tree data-structure of the BlobTree, denoted as A, and a set of time varying parameters, denoted as $\mathcal{P}_A(t)$.

The set of parameters $\mathcal{P}_A(t)$ includes the global time varying threshold parameter, denoted as $T_A(t)$, as well as the set of time varying parameters of the nodes $A_i(t)$ of the BlobTree, denoted as $\mathcal{P}_{A_i}(t)$:

$$\mathcal{P}_A(t) = \left\{ \begin{array}{c} T_A(t) \\ \mathcal{P}_{A_1}(t) = \{p_{A_{1,1}}(t), \ldots\} \\ \ldots \\ \mathcal{P}_{A_n}(t) = \{p_{A_{n,1}}(t), \ldots\} \end{array} \right\}$$

Therefore, $p_{A_{i,j}}(t)$ will denote the j^{th} time varying parameter of the i^{th} node of the BlobTree. Each parameter has its own evolution function which is independent of these of the other parameters.

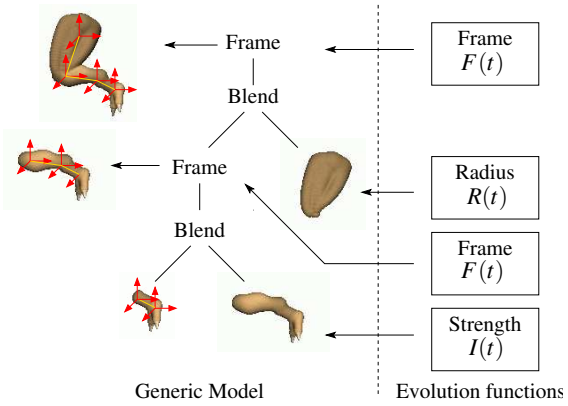

Figure 3: Closeup of the animated BlobTree model of the leg of the Tyrannosaurus-Rex character. Frame nodes are parameterized in time by quaternion splines, denoted as $F(t)$, that define the movement of the animation skeleton. The radius of some skeletal elements of the leg varies in time to create a bulging effect during the animation.

Splitting the animated BlobTree model into a generic structure and a set of parameters controlled by evolution functions (Figure 3) has a number of advantages. Since evolution functions may be of any type, our model supports a wide range of animation systems. Key-frame based animation systems, physically-based simulations and even comportemental methods may be used to characterize the time varying parameters. Time varying deformations, as well as metamorphoses, may be achieved in the same way.

The animation, deformation and metamorphosis of our model will be performed by creating instances of the generic BlobTree structure A using the parameters $\mathcal{P}_A(t)$ at a given time step t. The creation of those instances is achieved by recursively traversing the nodes of the generic BlobTree structure and computing the values of the parameters of the nodes with the evolution functions $p_{A_{i,j}}(t)$.

2.3 Control of the model through time

The next paragraphs review the different control parameters $\mathcal{P}_A(t)$ of the animated BlobTree.

Skeletal primitives Skeletal primitives are characterized by the parameters of their field function and skeleton. Field function parameters include the radius of influence $R(t)$, the maximum intensity $I(t)$ and the stiffness coefficient $k(t)$ of the potential function (Figure 4). The skeletons involve different geometrical parameters, depending on their type.

Polytopes are defined by their vertices $\mathbf{p}_i(t)$ whose coordinates may be functions of time. Cylinder, cone, disc, circle, and sphere primitives are characterized by their intrinsic time varying parameters, *i.e.* center or axis, and radius. The animation of spline curves and Coons patches is achieved by directly modifying the control points and normal vectors of the corresponding meta-primitives as we will see below.

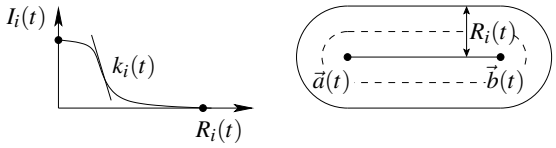

Figure 4: Time varying parameters of a skeletal primitive.

Blending and Boolean operators Boolean operators (union, intersection and difference) as well as the blending operator are not parameterized and therefore kept unchanged through-out an animation. The hyper-blending operator however may be parameterized by the power coefficient $q(t)$. Let us recall that hyper-blending operator proposed in [Wyvill et al. 1999] is a generalization of the Euclidean distance, using $\mathcal{L}^q$ metrics:

$$f(\mathbf{p}) = \left(\sum_{i=1}^{i=n} f_i(\mathbf{p})^q \right)^{1/q} \qquad q \in \mathbb{R}^*$$

As shown in figure 5, the hyper-blending operator may be used to produce smooth metamorphosis between blending and union by transforming the parameter $q(t)$ in interval $[1, +\infty[$.

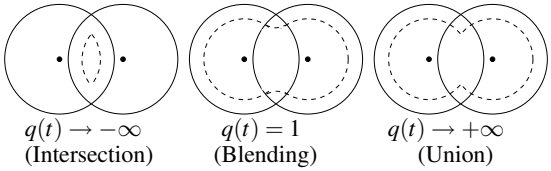

Figure 5: Evolution of the hyper-blending degree.

Warping operators The BlobTree includes warping operators that distort the shape of the implicit surface by warping space in its neighborhood. A warp is a homeomorphic function $\omega(\mathbf{p})$ that maps $\mathbb{R}^3$ into $\mathbb{R}^3$. The field function in space is defined as :

$$f\omega(\mathbf{p}) = f \circ \omega^{-1}(\mathbf{p})$$

In our implementation, we use the Barr operators [Barr 1984] and affine transformations. The twisting operator is parameterized by the end vertices of the twisting axis $\vec{a}(t)$ and $\vec{b}(t)$ and by the twisting angle $\theta(t)$. The tapering operator is also characterized by the end vertices of the tapering axis $\vec{a}(t)$ and $\vec{b}(t)$, and a time varying tapering function.

Affine transformation operators Although affine transformations are specific warps, they deserve special attention as they are involved in the definition of the animation skeleton.

Affine transformations are defined as a composition of scalings, rotations and translations. The whole sub-tree of a frame node is automatically affected by the evolution of the frame's parameters which allows a very simple and intuitive way to control a large set of skeletal primitives. Scalings and translations are simply characterized by two time varying vectors of coefficients. Rotations are defined using quaternion, and we use quaternion slerping [Möller and Haines 1999] to interpolate rotations through time so as to avoid any gimbal lock problem.

Meta-primitives Experiments show that the designer feels it uncomfortable to tune by hand the thousands of primitives of a complex model. Thus, we have defined the spline meta-primitive that freely combines a set of given nodes along any trajectory. For instance, the spiky backbone of the dragon model (Figure 13) has been designed as a spline meta-primitive of spikes. A specific case is a spline meta-primitive of spheres along a trajectory which approximate a spline curve. Likewise, the Coons patch meta-primitive generates the union of the desired number of triangles. Thus, the designer may define and finely control a large number of primitives whose animations are automatically generated from the animation of the meta-primitives through their vertices and normal vectors.

2.4 Discussion

The BlobTree is animated or deformed directly by defining the evolution functions of its nodes. Every single parameter of every node is a function of time, which provides a great flexibility and allows the animator to finely tune the animation if needed. The BlobTree metamorphosis and animorphosis also strongly relies on the parameterization of every node in the BlobTree model.

Apparently, one major drawback of this approach is that the number of evolution functions that need to be defined and controlled increases as the model becomes more complex. In practice, the hierarchical structure of the BlobTree effectively prevents from tuning every evolution function. As shown in figure 6, only a few dynamic parameters are needed to create convincing animations. Like for any hierarchical structure, the animated BlobTree profits by a hierarchical control over any transformation allowing precise control with a few tuning. Most of the evolutions of an animated model are solid movements of still nodes which are controlled with an unique frame node each. Thus, affine transformations play a major role in the animation, whereas twisting, tapering and bending nodes are responsible for the deformations of the model.

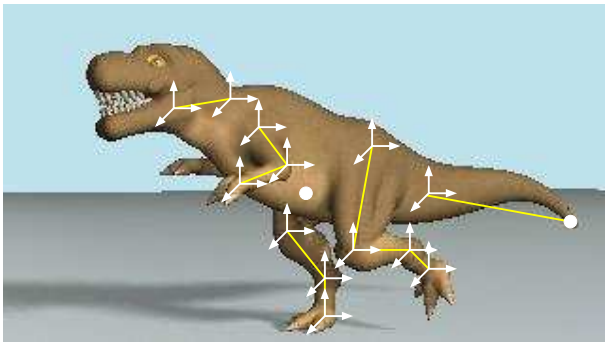

Figure 6: The animation skeleton: dots highlight dynamic parameters involving deformation (breathing and vertical tail movement) whereas frames outline the main articulations of the model.

Figure 6 shows the animation skeleton of our Tyrannosaurus-Rex over the still model. This BlobTree model includes 1835 skeletal primitives, 21 frame nodes and 201 boolean and blending operators for a total number of 2057 nodes. Thus, in theory, the model is parameterized by thousands of evolution functions. In practice however, most of parameters are kept constant as obtained from the still model. Only a few ones need to be defined to animate the Tyrannosaurus-Rex character. The movement of the legs, arms and head of the models are controlled by the hierarchy of the 21 frame nodes. The pulsating belly has been controlled by changing the radius of influence of the sphere primitives through time to fake breathing. The movement of the tail is directly controlled by 4 evolution functions that locate the control points of the spline meta-primitive through time.
Therefore, the overall animation is fully controlled by less than 30 evolution functions.

Hence, the control of the animation of the geometrically complex model stays both simple and fine. Moreover, since the movements are local, the model does not explode during an animation except if the animator wishes it, as shown in figure 14. Thus, during an animation, the volume of the model apparently stays the same and no artefacts appear.

Finally, the animation skeleton is internal to the structure of an animated BlobTree which is very beneficial to define the metamorphosis of complex models as we will see in the next section.

3 Animorphosis

In this section, we present the animorphosis of the BlobTree that aims at creating a smooth animated metamorphosis between two freely animated 3D-objects. This problem is more complex than the metamorphosis between still shapes. In general, a metamorphosis between two still objects is visually appealing and convincing if shape coherence is preserved during the transformation. In the context of animorphosis, both shape and animation coherence should be preserved. For instance, the transformation between a walking Tyrannosaurus-Rex and a flying Dragon should not only create smooth interpolating shapes. The animation of the legs of the Tyrannosaurus-Rex should smoothly transform into the slow balancing movement of the legs of the Dragon. The wings should not only progressively grow from its back but should also start fluttering with an increasing amplitude as the Dragon takes-off.

3.1 General algorithm

Let $A(t) = \{A, \mathcal{P}_A(t)\}$ and $B(t) = \{B, \mathcal{P}_B(t)\}$ denote the source and target models. The time varying parameters $p_{A_{i,j}}(t)$ and $p_{B_{i,j}}(t)$ of the nodes A_i and B_j will be denoted as $p_A(t)$ and $p_B(t)$ for the sake of clarity. Without loss of generality, we will assume that animorphosis takes place over the interval of time $[0,1]$.

We characterize the transformation by a new *generic* animated BlobTree model $C(t) = \{C, \mathcal{P}_C(t)\}$ defined as follows. The generic structure of C is computed by invoking the original BlobTree metamorphosis algorithm [Galin et al. 2000] between the corresponding generic models A and B. This generic structure will characterize the whole transformation. The time varying parameters $\mathcal{P}_C(t)$ should interpolate the parameters $\mathcal{P}_A(0)$ and $\mathcal{P}_B(1)$. This step is the most difficult, since a tight control is required to preserve both shape and animation coherence. The overall algorithm may be outlined as follows :

1. Define a graph of correspondences $\mathcal{G}_C(A \rightarrow B)$ matching two models A and B.
2. Create the generic structure C from the bijectively matching graph derived from $\mathcal{G}_C(A \rightarrow B)$.
3. Define each time varying parameter $p_C(t)$ by interpolating its corresponding parent parameters $p_A(t)$ and $p_B(t)$.

Creation of the generic tree structure The first two steps of the algorithm are performed as follows [Galin et al. 2000]. Given the tree structures A and B, we aim at creating two new overlapping models whose nodes and leaves can be bijectively paired. This involves the creation of a graph of correspondences compatible with

the tree structure of both the source and the target animated models (Figure 7).

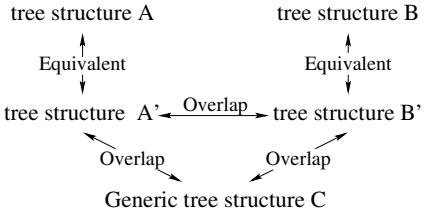

Figure 7: Generic tree structure creation process

Starting from the roots of the models, we iteratively descend down the tree structures and create a graph of correspondences between the nodes at the same level. Whenever a node of A holds several correspondence links, it is split into sub-nodes and the animated BlobTree $\{A, \mathcal{P}_A(t)\}$ is updated into an equivalent form denoted as $\{A', \mathcal{P}_{A'}(t)\}$. The same process is simultaneously performed on the nodes of B.

The correspondence process ends at the leaves of either structure. Multiply matched leaves of the structure are split as described in [Galin and Akkouche 1996]. In practice, this correspondence process may be either controlled by the animator or achieved automatically through heuristics.

The *generic* structure C is implicitly created during the correspondence and the decomposition steps. It is the same as the overlapping structures A' and B'.

Controlling the generic parameters The last step of the algorithm controls the way parameters change through time. The new animated BlobTree $C(t)$ should not only interpolate the initial and final shapes $A(0)$ and $B(1)$ but also produce a visually coherent interpolation between the animations of $A(t)$ and $B(t)$ taking into account both animations. This means that the animation of $A(t)$ should smoothly disappear while the animation of $B(t)$ takes place. Therefore, we are confronted with both shape and animation coherence.

As we have seen is the previous section, the animation skeleton is internal to the structure of an animated BlobTree. In fact, each parameter of the model is twice a geometric and an animation parameter ensuring the coherence of these two aspects during whatever evolution. Thus, interpolating the geometries constrains the animation of the intermediate shape and interpolating the animations constrains its geometry. We use an evolution function, denoted as $s(t)$, to interpolate and weight the influence of the parent parameters $p_A(t)$ and $p_B(t)$ of every parameter $p_C(t)$ through time :

$$p_C(t) = (1 - s(t))\, p_A(t) + s(t)\, p_B(t)$$

This dependency relation is a great advantage since our goal is to maintain the coherence between geometry and animation. In the example of the animorphosis of a walking Tyrannosaurus-Rex into a flying Dragon, the growth of the wings is so constrained by the evolution function defining the speed that the fluttering increases.

In practice, $s(t)$ may be of any type but should fulfill the requirements that $s(0) = 0$ and $s(1) = 1$. Altogether, the interpolation functions $s(t)$ control the animorphosis and are responsible for shape and animation coherence. In our implementation, $s(t)$ is a piecewise cubic spline that allows a fine and intuitive control of the transformation through the control of a few control knots. Figure 8 shows the control curve of the interpolation of the height parameters of the Tyrannosaurus-Rex and the Dragon. This curve constrains the model to the floor over the first third of the animorphosis interval and characterizes how rapidly the Dragon takes off.

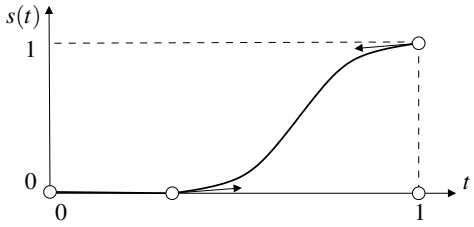

Figure 8: Control of the animorphosis of a parameter using a piecewise interpolation function $s(t)$. This evolution function constrains both the animation and the geometry evolutions of the model.

3.2 Discussion

As for animation, each time varying parameter of every node in the generic animated BlobTree model $C(t)$ may be characterized independently with its own evolution function. Although this approach provides a very tight control over the transformation, it requires the characterization of many parameters. Let us recall that only a few parameters of the source and target models are effectively dynamic. Here again, the alone parameters $p_C(t)$ whose one of their parent parameters is dynamic really have to be managed whereas the others may simply be characterized by the linear cross-dissolve of $p_A(t)$ and $p_B(t)$.

3.3 Key-frame animorphosis

In this section, we present our key-frame animation system and a semi-automatic method to define the animorphosis of two such animated shapes as an internal operator of the animated BlobTree model.

In our own key-frame animation system, the parameters of an animated BlobTree model are characterized by piecewise Hermite spline curves defined as follows. At the end times of an interval $[t^-, t^+]$, a parameter $\overline{p}$ is characterized by a column vector of its successive time derivatives. Let $p(t_i)$ represents the value of the parameter at time step t_i, $\dot{p}(t_i)$ and $\ddot{p}(t_i)$ represent respectively its speed and acceleration value at this same time. Thus $\overline{p}(t_i)$ is defined as follows :

$$\overline{p}(t_i) = \begin{pmatrix} p(t_i) \\ \dot{p}(t_i) \\ \ddot{p}(t_i) \\ \dots \end{pmatrix}$$

The value of $\overline{p}$ over the interval $[t^-, t^+]$ is given by the interpolation of Hermite-Ferguson of degree n which is a polynomial of degree $2n-1$.

We also use Hermite spline curves to define the interpolation functions $s(t)$ to control both shape and animation coherence. The designer may freely control the shape of the interpolation function by playing with the control points to produce the desired interpolation.

In this case, we can constrain the computation of the parameters $\overline{p}_C(t)$ so that animorphosis should produce a consistent animation model. The goal is to define the intermediate shape as a simple key-frame animated BlobTree rather than an interpolation of two animated BlobTrees at each time step. Given an interpolation function

$s(t)$, the following algorithm presents our semi-automatic method for controlling the animorphosis of two parameters $\overline{p}_A(t)$ and $\overline{p}_B(t)$ defined with evolution functions of this type (Figure 9) :

1. For each time interval $[t^-, t^+]$ of the interpolation function $s(t)$ defined by the animator, we define the set $\{t_i\}$ of the time steps of $\overline{p}_A(t)$ or $\overline{p}_B(t)$ in $[t^-, t^+]$.

2. For each t_i, we define a key step of $\overline{p}_C(t)$ as follows :

$$\overline{p}_C(t_i) = (1 - s(t_i))\overline{p}_A(t_i) + s(t_i)\overline{p}_B(t_i)$$

   The key steps of $p_C(t)$ at the end times of the control interval $[t^-, t^+]$ are obtained by plugging t^- and t^+ into the previous equation, thus :

$$\overline{p}_C(t^-) = (1 - s(t^-))\overline{p}_A(t^-) + s(t^-)\overline{p}_B(t^-)$$
$$\overline{p}_C(t^+) = (1 - s(t^+))\overline{p}_A(t^+) + s(t^+)\overline{p}_B(t^+)$$

3. We define the piece of the evolution function of $\overline{p}_C(t)$ by the interpolation of Hermite-Ferguson of the new key-steps $\{\overline{p}_C(t_i)\}$.

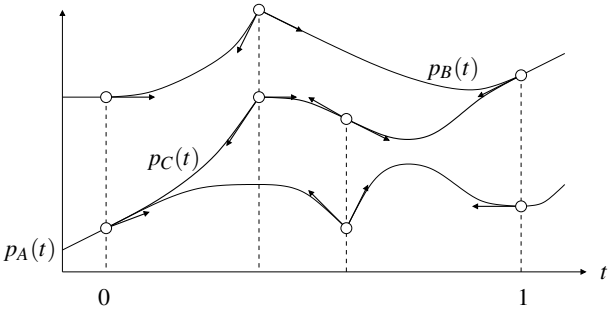

Figure 9: Definition of the intermediate parameter $\overline{p}_C(t)$ as a simple key-frame animated parameter whose time step values are defined as the constrained interpolation of the values of $\overline{p}_A(t)$ and $\overline{p}_B(t)$ at every of their time steps.

This technique creates at most $n_A + n_B + n_s$ key-frames for the parameter $\overline{p}_C(t)$ and enables a tight control of the overall evolution by sticking to the control points of the evolution of $\overline{p}_A(t)$ and $\overline{p}_B(t)$. This algorithm is applied to each parameter $\overline{p}_C(t)$. It defines an intermediate shape whose animation is characterized by a classical key-frame method and takes into account each time step of both the source and target models. Moreover, if necessary, extra control points may be inserted. It is generally the case for some of the important parameters, *i.e.* the few parameters which really are dynamic. This choice of the intermediate time steps ensures that both shape and animation coherence are preserved.

4 Results

Since we rely on a *generic* animated model that characterizes the whole transformation, the computation time needed to create all the instances through time is very fast and negligible compared to the rendering time: almost one thousand complex objects may be created in less than one second on a 750Mhz Duron. In comparison, displaying an instance with a few thousands of primitives takes ten to twenty seconds with a mesh and a couple of minutes in raytracing. We here present the different kinds of animorphosis depending on the way we use to animate the source and target models.

4.1 Key-frame animorphosis

Figure 11 shows a walking Tyrannosaurus-Rex controlled by a key-framing animation system. This model has been created with 2057 components including 1835 primitives, 21 frame nodes organized in a hierarchy and 201 boolean, blending and warping operators. The skeletons used for these primitives are vertices, spheres, cylinders and cones.

In practice, the modeling process is simpler than it seems thanks to the meta-primitives we use to arrange the elementary primitives. Thus, we only have had to tune by hand 66 spline meta-primitives to arrange 1788 of the 1835 primitives of this model. For the animation of the Tyrannosaurus-Rex, we have defined the animation skeleton of the still model and added the corresponding time varying frame nodes in the tree data-sructure. Then we have chosen a few others parameters to be controlled independently so as to perform specific animation effects such as the pulsating belly. The animation of the tail of the Tyrannosaurus-Rex has been simply defined by modifying the parameters of the control points of the underlying spline skeletal elements. It combines a rotation frame node that creates a left to right balancing movement synchronized with the hips. The vertical undulations are produced by directly moving the control vertices of its spline skeletal meta-primitives. Finally, the overall animation is controlled by only 30 parameters that are highlighted on the figure 6.

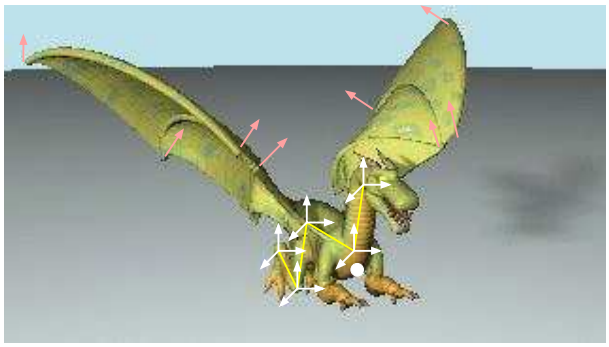

Figure 10: The animation skeleton of the Dragon : pink vectors highlight the control parameters of the Coons patch meta-primitives.

Figure 13 shows a flying Dragon. This model contains 5564 components including 5337 primitives, 27 frame nodes organized in a hierarchy and 200 boolean, blending and warping operators. The body of the Dragon has been created with point, sphere and cone skeletal elements sometimes arranged with spline meta-primitives. The wings have been created with triangles generated by Coons patch skeletal meta-primitives. Here again, only 51 parameters are used to control the movement of the model (Figure 10).

Figure 12 shows the transformation of the walking Tyrannosaurus-Rex into the flying Dragon. The correspondences graph between the two models is relatively simple since the initial and final shapes share the same kind of topology. The Dragon model needs more frame nodes due to the wings and the fine articulations of the forelegs and back legs.

The transformation of the animations is performed as follows. The animated wings grow faster than other components so that the Tyrannosaurus-Rex should take-off only when the wings get large enough.

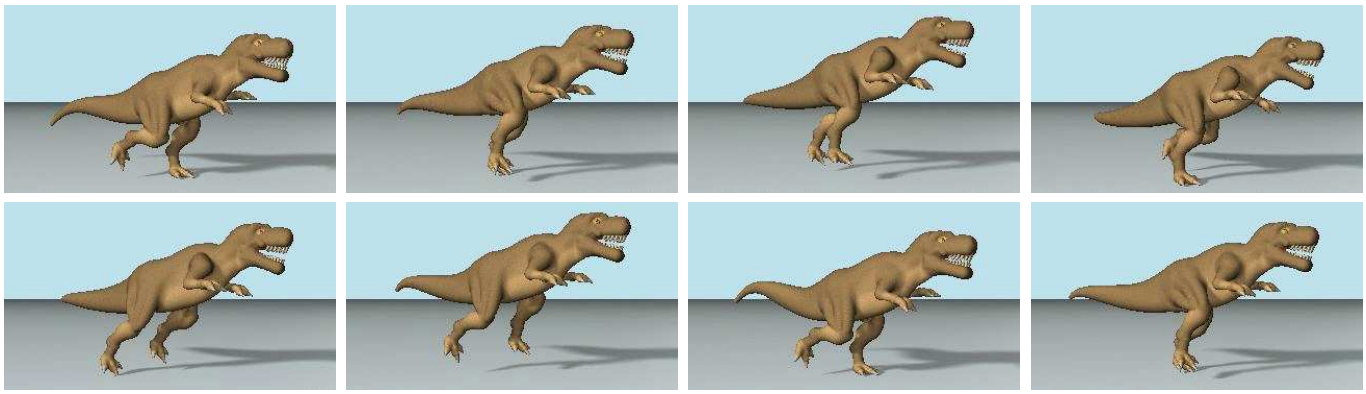

Figure 11: A walking Tyrannosaurus-Rex (key-frame animation).

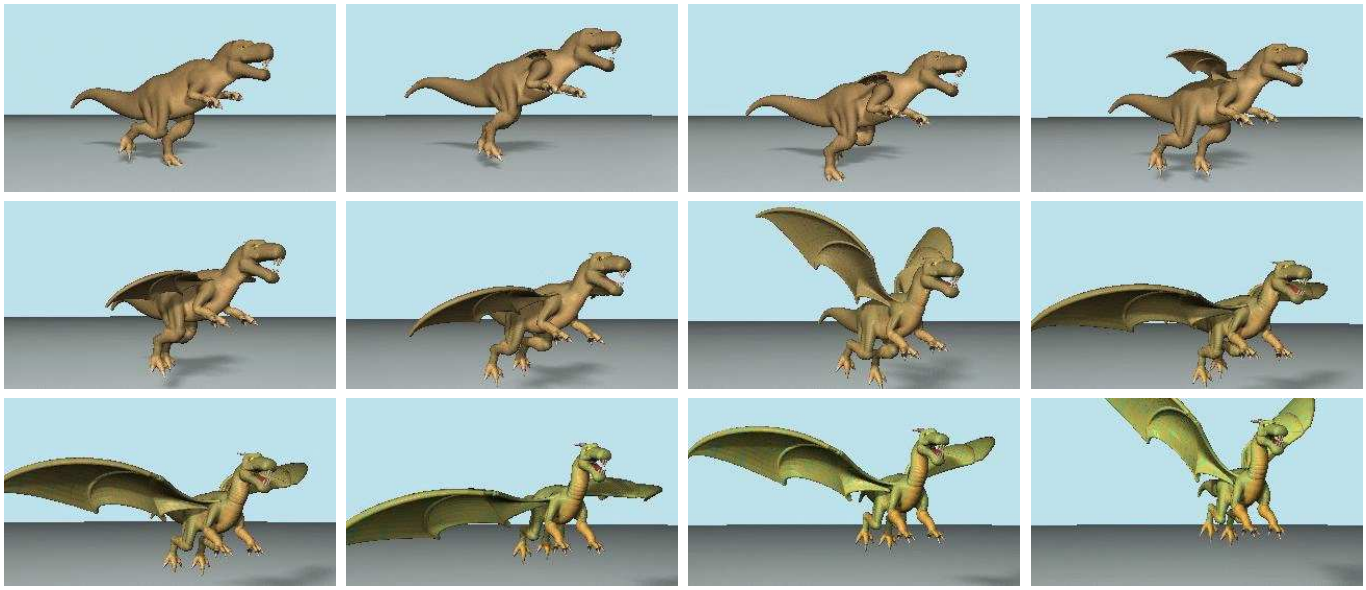

Figure 12: Animorphosis of a walking Tyrannosaurus-Rex into a flying Dragon (key-frame animorphosis).

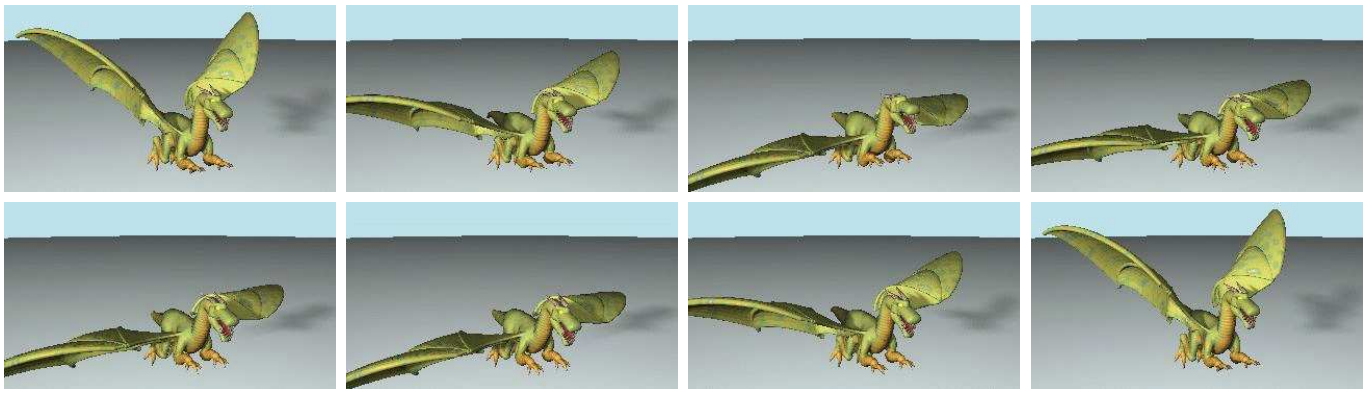

Figure 13: A flying Dragon (key-frame animation).

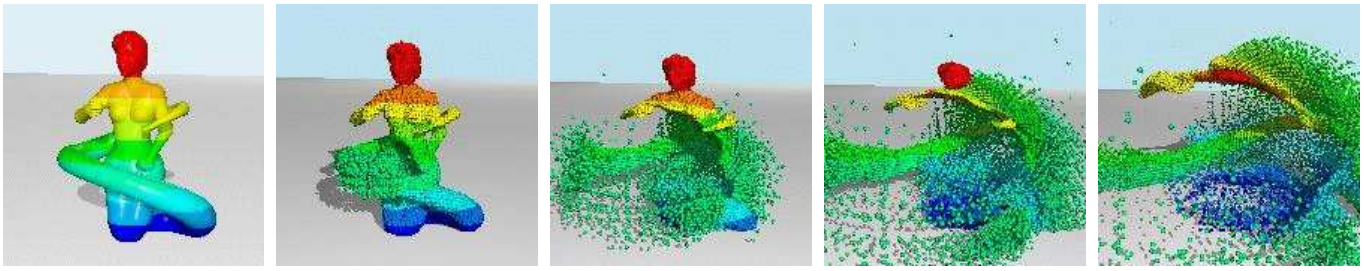

Figure 14: Explosion of a statue (physically-based animation).

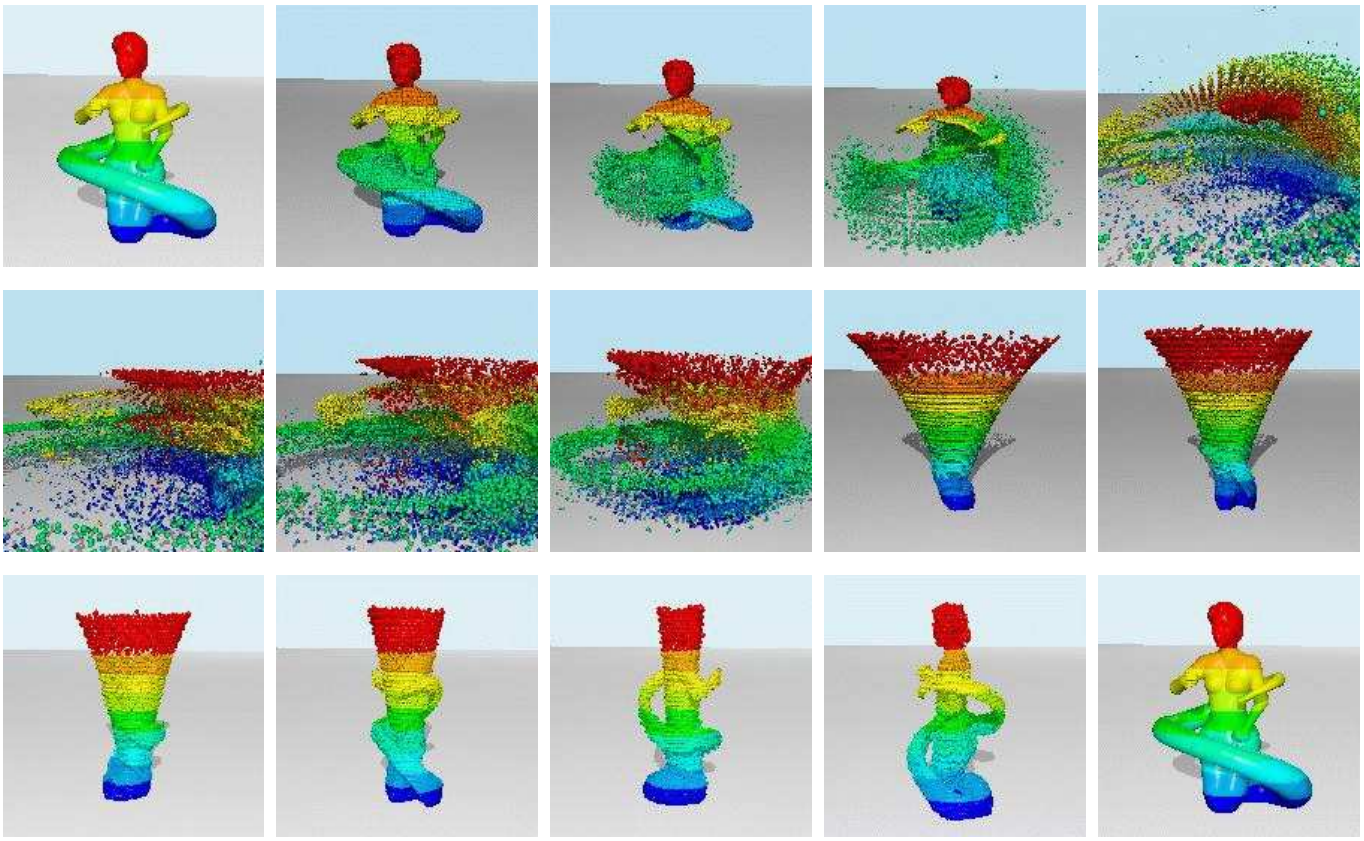

Figure 15: The statue explodes into pieces which smoothly form a swirling vortex. These particles eventually gather to recreate the original statue (physically-based animorphosis).

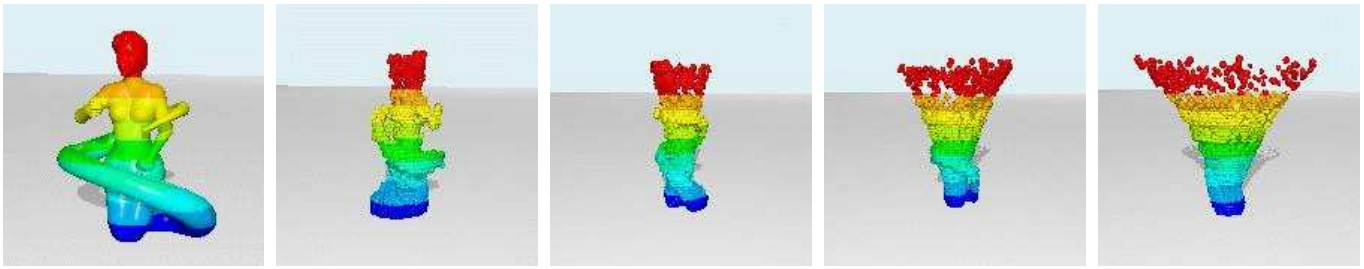

Figure 16: Disintegration of a statue into a swirling vortex of particles (physically-based animation).

The metamorphosis between the heads of the two models required a tighter control. The heads and necks were treated separately because of the transformation of the necks which is volumetrically more important. The jaws were controlled independently to avoid blending during their metamorphosis. It took us a couple of hours to create the graph of correspondences between the two models. Much time was spent finely matching the jaws and the different parts of the skulls to ensure shape coherence. Another couple of hours were necessary to tune the important interpolation parameters. In prtactice, most of the time was spent waiting for the renderings.

4.2 Physically-based animorphosis

Figure 15 represents a complex animorphosis of physically-based animations. A statue disintegrates into a set of particles because of the energy of an explosion. The trajectories of those particles are then animorphed with the inverse animation of a swirling vortex in order to re-build the statue.

Figure 17: A snake-woman statue.

The physically-based animorphosis was performed as follows. We first created the statue model (Figure 17) that contains 8862 nodes, including 8600 primitives and 262 boolean and blending operators. Here again, the modeling process has been simplified by the use of our meta-primitives. This time however, the animation is not performed by modifying the parameters of these meta-primitives. Instead, the statue was transformed into a set of 36100 particles. The animation of the explosion (Figure 14) was performed as prescribed in [Mazarak et al. 1999; Yngve et al. 2000] whereas the animation of the vortex (Figure 16) relies on a force field that constrains particles to a swirling movement.

5 Conclusion and future work

In this paper, we have presented a general framework to create and control metamorphoses between animated shapes. Our animated BlobTree model encompasses animation and metamorphosis in a unified and coherent fashion.

This model is characterized by a generic BlobTree structure whose nodes are controlled by arbitrary functions of time. Since evolution functions may be of any type, our model supports whatever animation system. Key-frame based animation systems, physically based simulations and even comportemental methods may be used to define the time varying parameters.

We have proved that our animated BlobTree allows the creation of a wide range of special effects easilly. The animorphosis generates convincing and visually appealing transformations between complex objects while preserving both shape and animation coherence. The creation of a generic model is fundamental. This property enables us to use the same accelerated techniques to ray-trace or polygonize any instance of the animated BlobTree model. In this context, the use of level of details in the design of complex BlobTrees could speed up rendering. Specific polygonization techniques that take advantage of temporal coherence will be needed in the creation of an interactive animorphosis editor.

A major challenge in implicit surface metamorphosis is the control of the blend between the different skeletal parts of the models. As prescribed in [Galin and Akkouche 1996], our method relies on the control of the trajectories of the skeletal elements to avoid blobby shapes. The local blending techniques proposed in [Angelidis et al. 2002; Pasko et al. 2002] could be incorporated in our implementation as a future work.

6 Acknowledgments

We would like to thank Tiphaine Accary who modelled the beautiful snake-woman statue in a few hours.

References

ALEXA, M. 2001. Mesh morphing. *Proceedings of Eurographics 2001 20*, 3 (September).

ANGELIDIS, A., JEPP, P., AND CANI, M.-P. 2002. Implicit modeling with skeleton curves: Controlled blending in contact situations. In *Shape Modeling International*, IEEE Computer Society Press, ACM. Banff, Alberta, Cananda.

BARR, A. H. 1984. Global and local deformations of solid primitives. *Computer Graphics Proceedings 18* (July), 21–30.

BLOOMENTHAL, J., BAJAJ, C., BLINN, J., CANI-GASCUEL, M.-P., ROCKWOOD, A., WYVILL, B., AND WYVILL, G. 1997. *Introduction to Implicit Surfaces*. Morgan Kaufmann.

COQUILLART, S. 1990. Extended free-form deformation: A sculpturing tool for 3D geometric modeling. *Computer Graphics 24*, 4, 187–196.

DESBRUN, M., AND CANI, M.-P. 1995. Animating soft substances with implicit surfaces. *Computer Graphics Proceedings 29* (August), 287–290.

FAUSETT, E., PASKO, A., AND ADZHIEV, V. 2000. Space-time and higher dimensional modeling for animation. *Computer Animation 2000* (May), 140–145.

FOSTER, N., AND FEDKIW, R. 2001. Practical animation of liquids. *Computer Graphics Proceedings*, 15–22.

FOX, M., CALBRAIGHT, C., AND BRIAN. 2001. Efficient implementation of the blobtree for rendering purposes. *Proceedings of Shape Modelling International, Genova, Italy* (May).

GALIN, E., AND AKKOUCHE, S. 1996. Soft object metamorphosis based on minkowski sums. *Proceedings of Eurographics'96 15*, 3 (August), 143–153.

GALIN, E., LECLERCQ, A., AND AKKOUCHE, S. 2000. Morphing the blobtree. *Computer Graphic Forum 19*, 4 (November), 257–270.

GASCUEL, M.-P. 1993. An implicit formulation for precise contact modeling between flexible solids. *Computer Graphics Proceedings* (August), 313–320.

HE, T., WANG, S., AND KAUFMAN, A. 1994. Wavelet-based volume morphing. *Proceedings of Visualization'94*, 85–91.

HUGHES, J. F. 1992. Scheduled fourier volume morphing. *Computer Graphics Proceedings 26*, 2, 43–46.

LAZARUS, F., AND VERROUST, A. 1998. Three-dimensional metamorphosis: a survey. *The Visual Computer 14*, 8/9, 373–389.

LAZARUS, F., COQUILLART, S., AND JANCÈNE, P. 1994. Axial deformations: an intuitive deformation technique. *Computer-Aided Design 26*, 8 (August), 607–613.

LEE, A., DOBKIN, D., SWELDENS, W., AND SHRÖDER, P. 1999. Multiresolution mesh morphing. *Computer Graphics Proceedings* (August).

LERIOS, A., GARFINKLE, C. D., AND LEVOY, M. 1995. Feature-based volume metamorphosis. *Computer Graphics* (August), 449–456.

MACCRACKEN, R., AND JOY, K. I. 1996. Free-form deformations with lattices of arbitrary topology. *Computer Graphics Proceedings 30*, 181–188.

MAZARAK, O., MARTINS, C., AND AMANATIDES, J. 1999. Animating exploding objects. *Graphics Interface*.

MILLER, G., AND PEARCE, A. 1989. Globular dynamics: a connected particle system for animating viscous fluids. *Computer and Graphics 13*, 3, 305–309.

MÖLLER, T., AND HAINES, E. 1999. *Real-Time Rendering*. A K Peters, Natick, Massachusetts.

NUR, M. A., LANG, K., WYVILL, B., AND BOURNE, G. 2001. Animating the escape response of stomphia coccinea from dr-masterias imbricata modeled using implicit surfaces. *Proceedings of Computer Graphics International, Hong-Kong*, 81–88.

PASKO, A., AND SAVCHENKO, V. 1995. Constructing functionally defined surfaces. *Proceedings of Implicit Surfaces'95*, 97–106.

PASKO, G., PASKO, A., IKEDA, M., AND KUNII, T. 2002. Bounded blending operations. In *Shape Modeling International*, IEEE Computer Society Press, ACM. Banff, Alberta, Cananda.

PLATT, J., AND BARR, A. H. 1988. Constraints methods for flexible models. *Computer Graphics 22*, 4, 279–288.

SEDERBERG, T., AND PARRY, S. 1986. Free-form deformation of solid geometric models. *Computer Graphics Proceedings 20*, 4, 151–160.

TERZOPOULOS, D., PLATT, J., BARR, A. H., AND FLEISHER, K. 1988. Elastically deformation models. *Computer Graphics 22*, 4, 205–214.

TONNESEN, D. 1991. Modelling liquids and solids using thermal particles. *Graphics Interface*, 255–262.

WYVILL, G., PHEETERS, C. M., AND WYVILL, B. 1986. Data structure for soft objects. *The Visual Computer 2*, 4, 227–234.

WYVILL, B., GUY, A., AND GALIN, E. 1999. Extending the csg tree (warping, blending and boolean operations in an implicit surface modeling system). *Computer Graphics Forum 18*, 2 (June), 149–158.

YNGVE, G., O'BRIEN, J., AND HODGINS, J. 2000. Animating explosions. *Proceedings of ACM SIGGRAPH, New Orleans* (July), 29–36.