Early Draft (preprint) of a paper presented at the Theory and Practice of Computer Graphics (TPCG) 2005.
The actual published paper contains better illustrations and presentations of some of the concepts.

Keyframing Particles of Physically Based Systems

Brent M. Dingle
Texas A&M University

John Keyser
Texas A&M University

Abstract
This paper will present a way to use keyframing methods of particle motion to enhance the visual effects and user controllability of physically based particle systems. This will be done using an adaptive correction methodology. This will allow for three general types of keyframing: position to position, density to density, and boundary to boundary. While similar techniques have been explored in flocking behaviors and robotic motion planning, this paper implements them in conjunction with physically based systems and allows a comparison of particle based keyframing to those achieved using other methodologies. To illustrate the technique we will present two examples. The first morphs between two particle images. The second forces a smoke-like substance to change into various letters of the alphabet. While these are specific examples the techniques presented herein should apply to most any particle based system to achieve a diverse range of effects.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling – Physically based modeling, I.3.6 [Computer Graphics]: Methodology and Techniques – Interaction techniques and I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism – Animation.

1 Introduction

In 1983 a method to model substances such as fire, water, clouds and smoke was presented in [Ree83]. One of the major advantages of this technique was the physical realism it allowed. However, human imagination is boundless and we often wish to create very artistic effects from such substances. For example we might wish to form letters from smoke, or form faces out of sand, or perhaps we simply wish to creatively dissolve images. Of course, we also wish to maintain a degree of physical realism in the process. Particle systems would seem to be a natural choice for achieving these artistic effects. However no general framework for realizing these effects has been presented for use with physically based particle systems.

Thus the goal of this paper is to present a general method to allow users to create artistic effects with particle systems, while maintaining a degree of physical realism. Specifically, we are proposing keyframing the position, velocity, density, orientation and color of the particles to be at a specific state at a specific time. Between keyframes and when no keyframe is active, the particles are subjected to physically based forces. The purpose of this is to allow more control over the visual effects produced by the particle system, while still maintaining a certain amount of physical realism.

We begin with a simple method of keyframing the position of every particle. We will then demonstrate how this basic method may be expanded to achieve more robust keyframing, such as density to density and boundary to boundary methods. As we present these methods, two examples of application will be illustrated. We will conclude with possible directions of further expanding these keyframing methods. For example by introducing a plausibility test of the generated paths.

The advantages offered by this technique are:

- Gains in user controllability.
- An increase in speed if implemented in parallel environments [Sim90] or on the graphics processing unit [KSW04].
- User controlled plausibility testing of the paths generated.
- Generality to be applied to most any particle based simulation.
- Particles move “naturally” until a keyframe becomes active.

2 Previous Work

Particle systems have been used in many ways. In computer graphics they often represent fuzzy or poorly defined objects such as gases and liquids [Ree83, ECP94, Sta99, YOH00, FSJ01]. They have also been used to model cloth [BHW94, BW98, CK02] and other deformable objects [TW88, CMN97, BCDD00]. Expanding upon that they can be used to model surfaces [PZVG00] and generate surface textures [Tur91].

While effects similar to those presented in this paper have been previously achieved, no general presentation of how they were achieved has ever been offered. There has been work done by others that might suggest such methods [ACM03], but none have specifically applied or discussed the methods with respect to physically based particle systems.

It is also noted that many rendering systems offer particle based effects. These systems often refer to keyframing particle effects. However, the keyframing they refer to does not involve the states of the individual particles, but rather the state and motion of the particle emitters. Effectively they are referring to sequencing the timing and positioning of particle effects. In a few of these systems it is possible to assign a goal for particles, however, it is only possible to assign one goal and other forces, specifically conditional forces, cannot be applied to the individual particles. Further the goals cannot be turned off or randomized for each particle. In the end, it may be possible to implement the keyframing methods we will present on existing rendering systems, however the methods themselves are not inherent to any such system.

Also related to this paper is the recent work on keyframing the motion of smoke [TMPS03, FL04] as well as more general animation techniques involving keyframing concepts [Ree81, SB85, Las87]. None of these applied keyframing techniques directly to particle states. So, while we are certainly using similar ideas, we will show how to apply them to particles in general.

3 Basic Method, Position to Position

The most straightforward method to keyframe particles is to specify each particle's initial and target position (or state). The difficulty is maintaining the natural behavior of the substance being modeled by the particles, while at the same time forcing it to do something very unnatural. With this in mind, if we are given an initial position and a final position for each particle and a set of external forces acting on the particle then it is possible to achieve an automatic 'in-betweening' of the keyframes using a simple process.

So, assume we are given the initial position of a particle, $\boldsymbol{p_0}$, which occurs at time t_0, and the next position of the particle $\boldsymbol{p_n}$, which occurs at time t_n. We are also given a constant time step of $\Delta t \leq t_n - t_0$, that will be used throughout the simulation and a list of forces that will act on the particle. With this information we are able to derive a force for each time step that will apply the given forces to the particle and guide the particle to its final destination. It should be noted that keyframes are not always active and we need not run from one keyframe to another, thus the particles may have periods of free motion.

For a given time step, i, let the current time be denoted t_i and let the sum of the forces acting on the particle be denoted by $\boldsymbol{F_i}$. For the same time step let $\boldsymbol{f_i}$ denote the force that if applied for the remaining time of $t_n - t_i = \Delta t_i$ would move the particle from its current position, $\boldsymbol{p_i}$, to its final location, $\boldsymbol{p_n}$, disregarding $\boldsymbol{F_i}$.

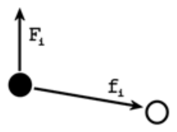

Figure 1: *Two calculated forces acting on a particle.*

This would give the total force acting on the particle to be: $(\boldsymbol{F_i} + \boldsymbol{f_i})\Delta t$. However, this would not guarantee that the particle would ever reach its target position $\boldsymbol{p_n}$. To make such a guarantee a scaling or weighting function, s_i, must be introduced. Thus the force acting on the particle would be: $(s_i \boldsymbol{F_i} + \boldsymbol{f_i})\Delta t$.

For simplicity we shall use a linear scaling function to define $s_i = \Delta t_i /(t_n - t_0)$. This is not the best function for all cases, and others could be used. This scaling term will automatically diminish the effect of $\boldsymbol{F_i}$. Notice the effect of $\boldsymbol{f_i}$ is inherently diminishing as the particle gets closer to its target, however, it may also be explicitly weighted if necessary. Further if we wanted each particle to behave differently a degree of randomness may be introduced by multiplying s_i by a random scalar $r_i \in (0, 1]$.

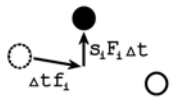

Figure 2: *Motion by scaled force sum for i^{th} time step.*

Having calculated the forces $\boldsymbol{F_i}$ and $\boldsymbol{f_i}$ and the scaling term s_i, we move the particle for the i^{th} time step by applying a force of $(s_i \boldsymbol{F_i} + \boldsymbol{f_i})\Delta t$.

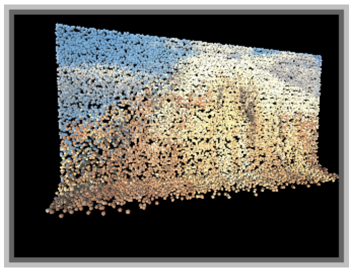

Figure 3: *An imaging dissolving under gravity.*

This position to position method was used to morph one image into another, as illustrated in figures 3 and 4. This morph began by initializing the particles so they displayed their image. They were then subjected to a variety of physically based forces, without any keyframing. After a certain amount of time passed, a keyframe became active guiding them back into a form that would display their contained image. Within this morphing a timely change of particle coloring was also implemented. This demonstrates a feature of this method by showing the particles need not move directly from one keyframe to another.

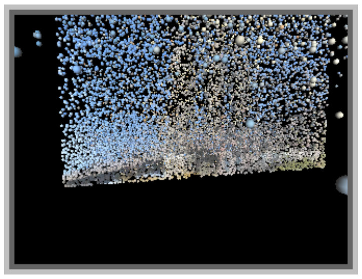

Figure 4: *An image reforming via keyframe forces.*

4. Density to Density Keyframes

Explicit position to position keyframes are limited in their use, because user specification of target goals for every single particle can become problematic. However there are many algorithms that employ the use of density and velocity functions, for example those used to model fluids or gases [EP90, Sak90, FM96, FSJ01]. The effects of these methods are impressive, however, they can take some time to simulate and render and do not inherently allow much control of the visual effect. Recent effort has been made to remedy this. For example, it is now possible to control the motion of smoke and similar substances [TMPS03, FL04] on grid based simulators. However, these were not particle based effects.

Because there are scenarios where the motion of the density of a substance is being considered it is useful to have the ability to keyframe particles based on area, or volume, densities. From a user perspective, this means moving a given number of particles from one area to another. Exactly which particles get moved where is no concern as long as the necessary number of particles ends up in the correct area. While this could be from multiple areas to multiple areas, for a simple presentation we will limit the scenario to a case of one source area and one or more target areas.

It should be emphasized that in this form of keyframing the shape of the source and target areas is of no concern. Only the number of particles is of any significance. Further while the word *area* is used in the descriptions the word *volume* equally applies. For additional simplicity we assume each mentioned area is of the same size. Thus densities are more directly identified by the number of particles in each area.

For density to density keyframing we are not concerned with which particles go where. Rather we want to move a given number of them from one area to another in the hope to simulate a density flow pattern. To achieve this the task is broken into three parts: particle selection, destination calculation and path generation. To implement this we say there exists a function for each part of the task. To demonstrate simple versions of these functions we make some assumptions. Among these are: there exist no inter-particle relationships and there is only one source area. If these assumptions are not true, more complicated functions may be used, and even if they are true, other functions may perform better in given scenarios. Yet the following worked for our scenarios.

Before creating the paths of particles involved in density to density keyframes it is necessary to identify which source particles will move to which target areas. Assuming we have only one source and one target area, this is trivial. If there is one source and two target areas, where the density of each target is half that of the source, it is easy to randomly select half of the source particles and assign them to the first target area and the other half to the second target area. This readily extends to three or more target areas and can be adapted to work if there are multiple source areas. However, if there are relationships between the particles themselves or other such considerations, then other selection methods may be used. Notice also that the source area must have a density great enough to support the target densities.

Once the source particles are selected and assigned to target areas it is necessary to determine where in the target area they will go. Obviously they could all go to the same location in the target area. However, that is usually not desired.

To determine the target location of each particle we begin with a coarse estimate based on a center of mass concept. If the particles are extremely dispersive in nature a refinement may be necessary.

To generate a coarse path we use a center of mass concept. To illustrate this consider the case where there is only one source area and one target area. The center of mass of the source area and target area is calculated. The simulation then runs forward. At each time step the center of mass of the particles is recalculated. The position to position keyframing is performed on the center of mass of the particles to obtain the $\boldsymbol{f_i}$ that will be applied to all the particles. The $\boldsymbol{F_i}$ is still unique to each particle. This allows a performance gain by reducing the number of $\boldsymbol{f_i}$ calculations, however if the particles are extremely dispersive in nature, or the relative size of areas involved is significantly different, the particles may not all end in the target area, which would require a refinement of the path. This need for refinement would require a detection method, such as the plausibility test described in section 6.

To refine the path we subdivide the particles based on their locations at each time step. This division is done using an adaptive kd-tree algorithm, where the divisions increase in number as the time approaches t_n. We then perform the same center of mass path planning as described above on each division. If necessary, this progresses down until each division contains only one particle. Thus, eventually, a path ending with the desired density in the target area is guaranteed.

This center of mass concept may be skipped. However doing so may slow the simulation. Should that not matter the easiest method for density to density keyframing is again to rely on randomness, and assign each source particle a random location in its target area. Once each particle is assigned a destination, its path is generated as described in the position to position keyframing. Again, if there are interparticle relationships to be maintained, or other restrictive considerations, then other target assignment methods may be used, but the method of path generation stays the same.

5. Boundary to Boundary Keyframes

Boundary to boundary keyframes are the most visually interesting of the methods. With this keyframing ability we can form many entertaining effects. Examples of such effects are illustrated in figures 5 and 7.

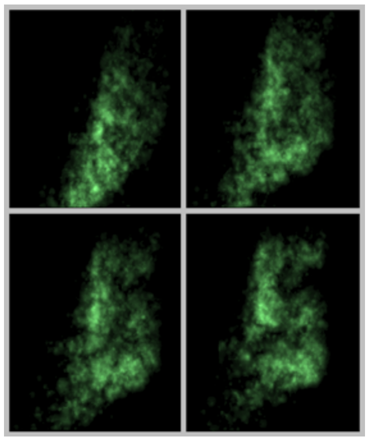

Figure 5: *The letter S formed by 6000 particles.*

For boundary to boundary keyframing we duplicate the processes used in density to density keyframing. However we remove the option of using a center of mass concept as we must guarantee the particles generate the desired shape. To do this we require a specific destination to be assigned to each particle based on a uniform random assignment of positions within the target boundary. More explicitly we again break the task into three parts: particle selection, destination calculation and path generation.

The particle selection method remains the same as in the density to density case. However the destination assignment while still random, must now be uniformly random within the target boundary. The path generation function remains the same, however, the center of mass concept can no longer be used. Thus $\boldsymbol{f_i}$ must be calculated for each particle.

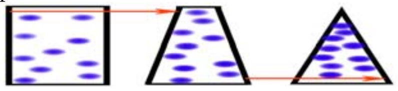

Figure 6: *A square morphing into a triangle.*

An expansion of this method also exists using current morphing techniques. Assuming the source and target boundaries are both closed then it is possible to create a morph between them [SG92]. From this we may obtain an intermediate boundary shape for each time step and we may confine the movement of the particles to stay within, or near, these intermediate boundaries. This confinement is enforced by the plausibility testing described in the next section. This usually produces a smoother transition.

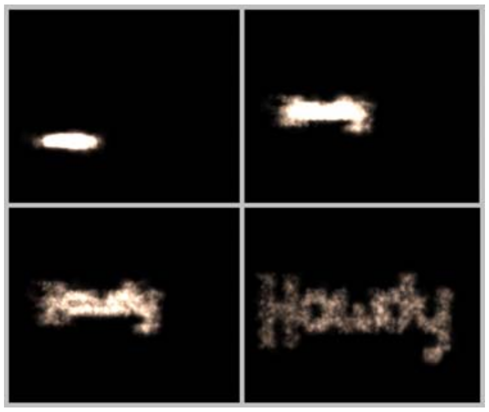

Figure 5: *Howdy forming from a ball of smoke.*

6 Plausibility

In all three keyframing methods there must be a concern for the plausibility of the paths the particles follow. Thus, once a path is generated its plausibility should be measured. This concept is well explained in [ACM03], and we will be modifying and adding to some of their results to be applied to physically based systems.

While we could approach plausibility as an optimality problem and apply techniques similar to those in [BN88, Coh92] for visual effects we do not necessarily want the most optimal solution and will likely desire some randomness to remain or be introduced in the motion. To accomplish this we will stray from the optimality methods and introduce a random scalar $r_i \in (0, 1]$ and offer a change of the equation presented in section 3, where the total force was: $(s_i\boldsymbol{F_i} + \boldsymbol{f_i})\Delta t$, it now becomes: $(r_i s_i\boldsymbol{F_i} + \boldsymbol{f_i})\Delta t$. Other randomization techniques may also be applied or specified by the user. Likewise the $\boldsymbol{f_i}$ term could be randomly scaled, however that removes the guarantee of hitting the target positions. This randomness allows multiple paths to be generated from the same algorithm. This should change the paths enough that some will be better than others.

It should be noted that technically the entire path from one keyframe to the next must be calculated to truly judge the plausibility of the path. To do this, speculative paths must be generated fast enough to not delay the visual display. However the activation and duration of keyframes is user specified. To reduce the runtime we do not always judge the plausibility of the entire path, but just a small subsection going only a few time steps ahead of the current time. The exact number of time steps is left as a parameter to the user. If that number amounts to a time greater than the largest active keyframe duration, then the plausibility of the entire path will be performed for all keyframes. While that should generate better paths it is not required and may slow the runtime performance.

The process of generating paths and testing their plausibility is performed until a user specified level of plausibility is achieved or a given number of attempts is exhausted.

6.1 Plausibility Criteria

For our simulation methods, the plausibility is a measure based on:

- *d* = the distance of particles from their target positions,
- *p* = the viability of the particle positions,
- *v* = the ratio of the magnitudes of the velocity of the particles between time steps.

This plausibility is comparative in nature so the first path generated will always be accepted, but may be replaced by successively generated paths. To express this we will follow notation similar to that presented in [AMC03]. However we will be testing individual particle paths, not all the paths all at once. So, letting g(*candidate path*) be the plausibility rating of a newly generated path and g(*current path*) be the plausibility rating of the currently chosen path then the probability of choosing the new path over the currently chosen path is:

$$P_{accept} = \mathrm{g}(\textit{candidate path}) \,/\, \mathrm{g}(\textit{current path})$$

This allows the new path to be chosen if P_{accept} is greater than a user specified value.

For a given path we will define three functions; g(*d*), g(*p*) and g(*v*) such that g(*path*) = g(*d*)*g(*p*)*g(*v*). The details of each of these functions is described below.

It is important to understand these are only suggested criteria. Other measures of plausibility may be used as needed. Notice also each plausibility test is across only a small number of time steps, possibly one or perhaps the entire time from initial state to keyframe state. The number of time steps being considered will determine how reliable the plausibility test is.

6.2 Distance Plausibility

The distance plausibility of a path, g(*d*) is a measure of how close the particles are to their target states. In density to density keyframes this may be applied to the center of mass rather than individual particles. In every method it is defined as:

$$\mathrm{g}(d) = \frac{1}{\sigma_d\sqrt{2\pi}}\, e^{-\left\|Pos(part[i]) - Dest(part[i])\right\| / 2\sigma_d^2}$$

where *part*[*i*] is the particle indexed by *i*, Pos(*part*[*i*]) is the current position of *part*[*i*], Dest(*part*[*i*]) is the target position of *part*[*i*] and σ_d is a small user defined constant, 0.1 to 0.5 should work. This is similar to the center of mass measure presented in [ACM03], though it is being used a little differently here. Other measures should also work.

6.3 Viability Plausibility

The plausibility of the viability of the ending particle position of a path is a measure of whether the particle can be or should be in that location. Thus it is defined as two functions:

$$\mathrm{g}(p) = h_c * h_s.$$

The "can be" part of the measure, h_c, is a Boolean function. If at any time of the path being considered, the particle is sitting somewhere that it cannot be, such as inside another object, $h_c = 0$, otherwise $h_c = 1$. Other criteria for this may be used, and the function need not be Boolean, however for our purposes this was sufficient.

The "should be" part of the measure, h_s, only applies in the case of a boundary to boundary morphing keyframe and is a function of the square distances of the particles from their temporary target locations. This is defined as:

$$h_s = e^{-k * sqrdist(part[i])}$$

where *k* is a user supplied constant and sqrdist(*part*[*i*]) is the distance squared from *part*[*i*] to its target destination. Values between 5 and 20 work well for *k*. This is similar to the shape measure presented in [ACM03]. However the h_c term is unique to this paper and our points of distance measure for h_s are different.

6.4 Velocity Plausibility

The plausibility of the magnitude of the velocity of the particles, g(*v*), is necessary to achieve a visual smoothness in motion. This measure is unique to this paper. For one time step

$$\mathrm{f}(v) = e^{\frac{-c * \left\|vel_c(part[i]) - vel_p(part[i])\right\|}{\left\|vel_p(part[i])\right\|}}$$

where vel_p(*part*[*i*]) is the velocity of *part*[*i*] on the previous time step, vel_c(*part*[*i*]) is the current velocity and *c* is a user defined constant. Values near 1 should work well for *c*.

From this g(v) is the product of all f(v) across all the time steps used to generate the path:

$$g(v) = \prod_{all\_time\_steps} f(v)$$

7. Conclusion and Future Work

We have now described three keyframing methods applicable to physically based particle systems. These methods should be easy to implement and incorporate in already existing systems. The methods may be combined to achieve a variety of effects. In such a combination the priority of each effect must be determined in the path generation. It is recommended to give position to position the highest priority and density to density the lowest. However it could be left to the user to specify.

The majority of the discussion above is dealing with just the position of the particles. This is mostly for understandability. The ideas presented can be applied to any state variable of the particles, such as color, transparency, rotation, velocity, etc.

In a similar fashion the criteria and weighting functions we have chosen are for demonstration, they are not the only choices. It might also be possible to achieve the same results using other methods. For example, while we are considering velocity directly as a plausibility criteria it may also be controlled by using smaller time steps or increasing the time between keyframes. However for consistency within this method we treat it as a plausibility criterion.

We note the visual appeal of particle effects does not meet everyone's standards. However, these are realtime effects, and are likely to be made faster with the advancement of GPU programming techniques. Further these methods need only be used to prototype an effect after which more advanced techniques and a larger amount of time could then be dedicated to final renderings.

In conclusion, it should be obvious that implementing this method would allow for greater user control of physically based particle effects. While the concept of plausibility has been presented by others, we have shown that it can work within this general, physically based framework. Further, we have offered a way the user may control how well the plausibility tests perform. This is allowed not only by setting the parameters of tests, but also by setting for what time length of a path they will be applied. In all of this we have been integrating the usage of keyframe constraints with physical forces. There is no reason these forces need to be reality based and the method should work for any set of external forces or rules of motions. *Of importance is that the particle motion is not always keyframed; the particles may behave "normally" until a keyframe becomes active.* Implementing such a method will allow for a myriad of effects to be obtained not currently attainable in as easy of a fashion.

Acknowledgements

We would like to thank Dr. Donald House of the Visualization Sciences program at Texas A&M University for his support and guidance in the development of this paper. We would also like to thank David Eberle for his encouragement, and the authors of [ACM03] for inspiration.

References

[AMC03] Anderson, Matt and McDaniel, Eric and Chenney, Stephen, Constrained Animation of Flocks, Eurographics Symposium on Computer animation, pp. 286-297, San Diego, 2003.

[BCDD00] Barr, A., Cani, M.P., Debunne, G., Desbrun, M., Adaptive simulation of soft bodies in real-time, Computer Animation 2000 Proceedings, pp. 15-20, 2000.

[BHW94] Breen, D.E., House, D.H. and Wozny, M.J., Predicting the Drape of Woven Cloth Using Interacting Particles, SIGGRAPH'94, pp. 365-372, 1994.

[BN88] Brotman, Lynne Shapiro and Netravali, Arun N., Motion Interpolation by Optimal control, Computer Graphics, Vol. 22, No. 4, August, 1988.

[BN92] Beier, Thaddeus and Neely, Shawn, Feature-based image metamorphosis, SIGGRAPH '92, pp. 35-42, 1992.

[BW98] Baraff, Witkin, Large steps in cloth simulation, SIGGRAPH 1998.

[Coh92] Cohen, Michael F., Interactive Spacetime Control for Animation, Computer Graphics, Vol. 26, No. 2, July 1992.

[CK02] Choi, Kwang-Jin and Ko, Hyeong-Seok, Stable but Responsive Cloth, SIGGRAPH 2002, Vol. 21, No. 3, July 2002.

[CMN97] Christensen, J., Marks, J. and Ngo, J. T., Automatic motion synthesis for 3D mass-spring models, The Visual Computer, Vol. 13, pp. 20–28, 1997.

[ECP94] Ebert, D.S., Carlson, W.E., and Parent, R.E., Solid Spaces and Inverse Particle Systems for Controlling the Animation of Gases and Fluids, The Visual Comp., 10:179-190, 1994.

[EP90] Ebert, D. S. and Parent, R. E. Rendering and Animation of Gaseous Phenomena by Combining Fast Volume and Scanline A-buffer Techniques, SIGGRAPH'90, Vol. 24, No. 4, pp. 357-366, August 1990.

[FL04] Fattal, R. and Lischinski, D., Target-Driven Smoke Animation, SIGGRAPH'04, Los Angeles, Vol. 23, No. 3, pp. 264-270, August 2004.

[FM96] Foster, N. and Metaxas, D., Realistic Animation of Liquids, Graphical Models and Image Processing, Vol. 58, No. 5, pp. 471-483, 1996.

[FSJ01] Fedkiw,Ronald and Stam,Jos and Jensen, Henrik Wann, Visual Simulation of Smoke, SIGGRAPH'01, pp. 15-22, August, 2001.

[KSW04] Kipfer,Peter and Segal,Mark and Westermann, Rudiger, UberFlow: A GPU-Based Particle Engine, Eurographics 2004, Grenoble, France, 2004.

[Las87] Lassester, John, Principles of traditional animation applied to 3D computer animation, SIGGRAPH’87, pp. 35-44, 1987.

[OF02] Osher, S. and Fedkiw, R., Level Set Methods and Dynamic Implicit, Surfaces, Springer-Verlag, New York, 2002.

[PZVG00] Pfister, H., Zwicker, M. , van Baar, J., Gross, M., Surfels: Surface Elements as Rendering Primitives, SIGGRAPH 2000, pp. 335-342, July 2000.

[Ree81] Reeves, W., Inbetweening for Computer Animation Utilizing Moving Point Constraints, SIGGRAPH’81, pp. 263-270, 1981.

[Ree83] Reeves, W., Particle Systems: A Technique for Modeling a Class of Fuzzy Objects, Proc. SIGGRAPH’83, pp. 359-376, 1983.

[Sak90] G. Sakas. Fast Rendering of Arbitrary Distributed Volume Densities, Proceedings of Eurographics ’90, pp. 519-530, September 1990.

[SB85] Steketee, Scott N. and Badler, Norman I., Parametric keyframe interpolation incorporating kinetic adjustment and phrasing control, SIGGRAPH’85, pp. 255-262, 1985.

[SG92] Sederberg, Thomas W. and Greenwood, Eugene, A physically based approach to 2-D shape blending, SIGGRAPH’92, pp. 25-34, 1992.

[Sim90] Sims, K., Particle Animation and Rendering Using Data Parallel Computation, Computer Graphics SIGGRAPH’90, vol. 24, no. 4, pp. 405-413, 1990.

[Sta99] Stam, J., Stable Fluids, SIGGRAPH’99, pp. 121-128, 1999.

[TMPS03] Treuille, Adrien , McNamara, Antoine, Popovic, Zoran and Stam, Jos, Keyframe Control of Smoke Simulations, SIGGRAPH’03, Vol. 22, No. 3 pp. 716-723, July, 2003.

[Tur91] Turk, Greg, Generating Textures on Arbitrary Surfaces Using Reaction-Diffusion, Computer Graphics, Vol. 25, No. 4, pp. 289-298, July 1991.

[TW88] Terzopoulos, D. and A. Witkin, Physically-based models with rigid and deformable components, Proc. Graphics Interface, pp. 146-154, June, 1988.

[YOH00] Yngve, G. D., O'Brien, J. F., Hodgins, J. K., 2000, Animating Explosions. The proceedings of ACM SIGGRAPH 2000, New Orleans, July 23-28, pp. 29-36, 2000