EG UK Theory and Practice of Computer Graphics (2005)
L. Lever, M. McDerby (Editors)

Keyframing Particles of Physically Based Systems

Brent M. Dingle and John Keyser

Texas A&M University

Abstract

This paper will present a way to use keyframing methods for 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 keyframing 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, 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.

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 for further expanding these keyframing methods. For example we can introduce a plausibility test of the generated paths.

The advantages offered by this technique are:

- Gains in user controllability.
- Speed increases if implemented in parallel environments [Sim90] or on the graphics processing unit [KSW04].
- User controlled plausibility testing of the paths generated.
- May 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, DDCB00]. Expanding upon that they can be used to model surfaces [PZvBG00] 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 been offered. There has been work done by others that might suggest such methods [AMC03], but few 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 $\triangle t \leq t_n$ - 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 = $\triangle t_i$ would move the particle from its current position, $\boldsymbol{p}_i$, to its final location, $\boldsymbol{p}_n$, disregarding $\boldsymbol{F}_i$.

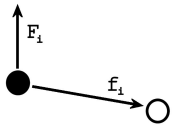

Figure 1: *Two calculated forces acting on a particle.*

This would give the total force acting on the particle to be: $(\boldsymbol{F}_i + \boldsymbol{f}_i)\ \triangle t$ 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: $(s_i\ \boldsymbol{F}_i + \boldsymbol{f}_i)\triangle t$.

For simplicity we shall use a linear scaling function to define $s_i = \triangle t_i\ /(t_n - t_0)$. 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 $r_i \in (0, 1]$.

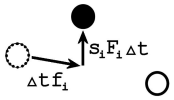

Figure 2: *Motion by scaled force sum for i^{th} time step.*

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 $(s_i\ \boldsymbol{F}_i + \boldsymbol{f}_i)\triangle t$.

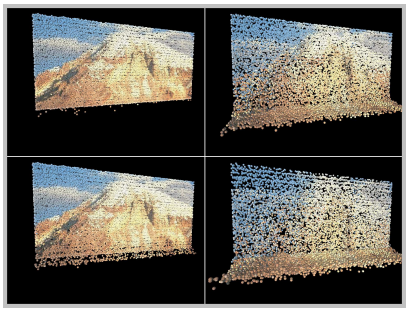

Figure 3: *An image dissolving under gravity.*

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.

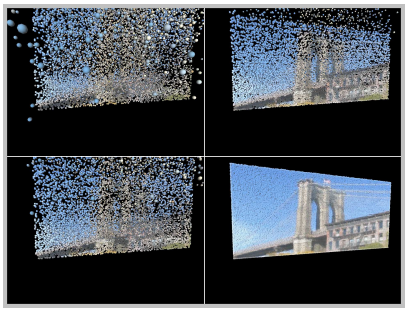

Figure 4: *An image reforming via keyframe forces.*

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 use a different function for each part of the task. We make some assumptions to demonstrate simple versions of these functions. 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 well 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.

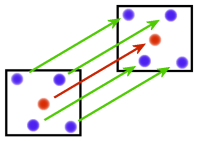

Figure 5: *In density to density keyframing, with no external forces, particles may move as the center of mass.*

To illustrate our approach 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.

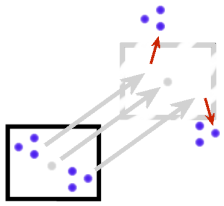

Figure 6: *Dispersive external forces may drive the particles too far apart.*

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.

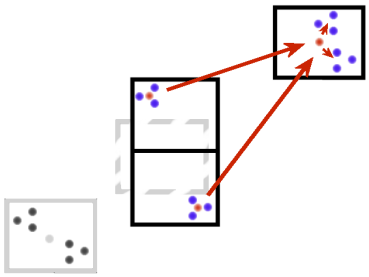

Figure 7: *Adaptive subdivision may correct the dispersion.*

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 8 and 10.

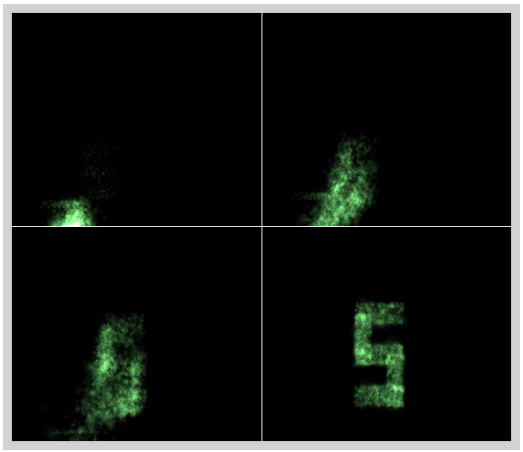

Figure 8: *The letter S formed from 6000 particles.*

For boundary to boundary keyframing the steps are the same as used in density to density keyframing. Specifically, we again break the task into three parts: particle selection, destination calculation and path generation.

For the particle selection process we again must select which particles from which source areas will go to which target areas. For ease of presentation we will assume there is only one source area and one target area. Should that not be the case random source to target area assignments may be performed, perhaps weighted by proximity to one another.

To calculate the destination of the particles we must emphasize that the particles are expected to visibly form the target area's shape. Thus they must fill the shape as much as possible. To accomplish this each source particle is assigned a specific target location. These target locations are generated as a uniform random sample within the target area.

As each particle has a specific destination generated for it, the path generation for each particle is identical in procedure to that described for position to position cases. Notice this means an $\boldsymbol{f}_i$ must be calculated for each particle.

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

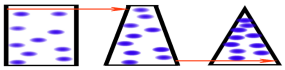

Figure 9: *A square morphing into a triangle.*

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.

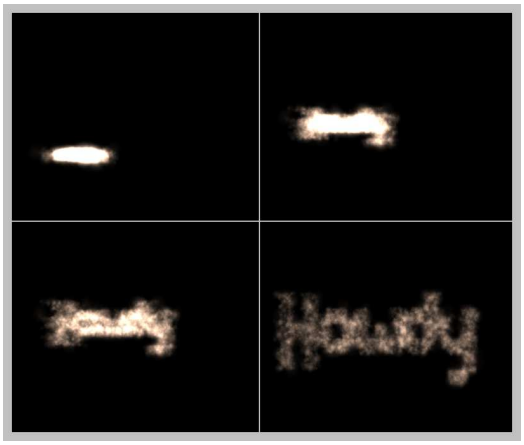

Figure 10: *Howdy forming from a ball of smoke.*

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 explained well in [AMC03], 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: $(s_i\boldsymbol{F}_i + \boldsymbol{f}_i)\triangle t$, it now becomes: $(r_i s_i\boldsymbol{F}_i + \boldsymbol{f}_i)\triangle t$. 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:

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

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:

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

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 s_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 [AMC03], though it is being used a little differently here. Other measures should also work. Such measures should have near zero values when the particle is far from its destination and go towards one as the particle nears its destination.

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:

$$g(p) = h_c * h_s.$$

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 [AMC03]. However the h_c term is unique to this paper and our points of distance measure for h_s are different. Other functions for h_s may be used. Such alternate functions work best if they are near zero most of the time and go sharply towards one as the particle nears its destination. The desired behavior of h_c and h_s is to strongly discourage "impossible" paths while still giving preference to paths that bring the particle closer to any intermediate destination it may have been assigned.

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

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

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. Other functions may be used for f(v) as long as they are near one when the magnitude of the change in velocity is near zero and tend towards zero as the magnitude tends to infinity.

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

$$g(v) = \prod_{all\ time\ steps} f(v)$$

This should discourage sudden, large velocity changes across time steps.

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 or selectively applied to achieve a variety of effects.

Most of the balance between keyframe control and physical control of the particles depends on the functions used to blend the forces as well as functions to determine the plausibility of the effects. These functions are expected to have parameters allowing the user to 'tweak' the effect as desired. While this paper presents examples of each of these functions others may be developed as needed. It is also important to note that each of the keyframes must allow the selection of which and how many particles it will effect. In this it is possible to move various sets of particles coming from the same generator in different ways. It also allows some particles to behave naturally while others are partially guided by keyframes. While all the particles may be keyframed, and thus all may be controlled by physical forces and keyframe forces, in development it was noted that allowing some of the particles to remain free from keyframing produced a more realistic feel to the animations. For example if only a portion of the particles of a smoke effect is used to form a letter, the observer tends to more readily accept the behavior. So even if some of the smoke is forming into a letter, because some of it is still "doing what it should," the observer tends to more willingly suspend reality.

It is also noted, 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, again, 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, most running at 30 FPS or better, 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 [AMC03] for inspiration.

The sample programs illustrated in this paper were run under the Windows XP operating system. The hardware consisted of a 3 GHz, Pentium IV processor, 512 MB of RAM, and an NVIDIA GeForce FX graphics card with 32 MB of memory. Each program was written using MS Visual C and OpenGL. The image morph shown in figures 3 and 4 is composed of approximately 2000, 3D spheres each with 16 slices and 16 stacks. It ran at 10 frames per second (FPS) or better. The smoke-like effect shown in figure 8 was generated by roughly 6000, 2D circles. It ran at 30 FPS or better. The effect shown in figure 10 was generated using around 12,000 particles and also ran at 30 FPS or better. As written, none of the programs were designed to specifically use any GPU functions. If such provisions were made significant gains in performance would likely be observed.

References

[AMC03] ANDERSON M., MCDANIEL E., CHENNEY S.: Constrained animation of flocks. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer animation* (2003), Eurographics Association, pp. 286–297.

[BHW94] BREEN D. E., HOUSE D. H., WOZNY M. J.: Predicting the drape of woven cloth using interacting particles. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1994), ACM Press, pp. 365–372.

[BN88] BROTMAN L. S., NETRAVALI A. N.: Motion interpolation by optimal control. *Computer Graphics 22*, 4 (August 1988).

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), ACM Press, pp. 43–54.

[CK02] CHOI K.-J., KO H.-S.: Stable but responsive cloth. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), ACM Press, pp. 604–611.

[CMN97] CHRISTENSEN J., MARKS J., NGO J. T.: Automatic motion synthesis for 3d mass-spring models. *The Visual Computer 13*, 1 (1997), 20–28.

[Coh92] COHEN M. F.: Interactive spacetime control for animation. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1992), ACM Press, pp. 293–302.

[DDCB00] DEBUNNE G., DESBRUN M., CANI M.-P., BARR A. H.: Adaptive simulation of soft bodies in real-time. In *Computer Animation 2000, Philadelphia, USA* (May 2000), pp. 133–144.

[ECP94] EBERT D. S., CARLSON W. E., PARENT R. E.: Solid spaces and inverse particle systems for controlling the animation of gases and fluids. *The Visual Computer 10*, 4 (1994), 179–190.

[EP90] EBERT D. S., PARENT R. E.: Rendering and animation of gaseous phenomena by combining fast volume and scanline a-buffer techniques. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1990), ACM Press, pp. 357–366.

[FL04] FATTAL R., LISCHINSKI D.: Target-driven smoke animation. *ACM Trans. Graph. 23*, 3 (2004), 441–448.

[FM96] FOSTER N., METAXAS D.: Realistic animation of liquids. *Graph. Models Image Process. 58*, 5 (1996), 471–483.

[FSJ01] FEDKIW R., STAM J., JENSEN H. W.: Visual simulation of smoke. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM Press, pp. 15–22.

[KSW04] KIPFER P., SEGAL M., WESTERMANN R.: Uberflow: a gpu-based particle engine. In *HWWS '04: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware* (New York, NY, USA, 2004), ACM Press, pp. 115–122.

[Las87] LASSETER J.: Principles of traditional animation applied to 3d computer animation. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1987), ACM Press, pp. 35–44.

[PZvBG00] PFISTER H., ZWICKER M., VAN BAAR J., GROSS M.: Surfels: surface elements as rendering primitives. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 335–342.

[Ree81] REEVES W. T.: Inbetweening for computer animation utilizing moving point constraints. In *SIGGRAPH '81: Proceedings of the 8th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1981), ACM Press, pp. 263–269.

[Ree83] REEVES W. T.: Particle systems: a technique for modeling a class of fuzzy objects. *SIGGRAPH Comput. Graph. 17*, 3 (1983), 359–375.

[Sak90] SAKAS G.: Fast rendering of arbitrary distributed volume densities. In *Proceedings of Eurographics '90* (September 1990), pp. 519–530.

[SB85] STEKETEE S. N., BADLER N. I.: Parametric keyframe interpolation incorporating kinetic adjustment and phrasing control. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1985), ACM Press, pp. 255–262.

[SG92] SEDERBERG T. W., GREENWOOD E.: A physically based approach to 2-d shape blending. In *SIGGRAPH* (1992), pp. 25–34.

[Sim90] SIMS K.: Particle animation and rendering using data parallel computation. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1990), ACM Press, pp. 405–413.

[Sta99] STAM J.: Stable fluids. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 121–128.

[TMPS03] TREUILLE A., MCNAMARA A., POPOVIć Z., STAM J.: Keyframe control of smoke simulations. *ACM Trans. Graph. 22*, 3 (2003), 716–723.

[Tur91] TURK G.: Generating textures on arbitrary surfaces using reaction-diffusion. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1991), ACM Press, pp. 289–298.

[TW88] TERZOPOULOS D., WITKIN A.: Physically based models with rigid and deformable components. *IEEE Comput. Graph. Appl. 8*, 6 (1988), 41–51.

[YOH00] YNGVE G. D., O'BRIEN J. F., HODGINS J. K.: Animating explosions. In *Proceedings of ACM SIGGRAPH 2000* (Aug. 2000), pp. 29–36.