ENHANCING THE CONTROL AND PERFORMANCE OF PARTICLE SYSTEMS THROUGH THE USE OF LOCAL ENVIRONMENTS

Daniel O. Kutz
Richard R. Eckert
State University of New York at Binghamton
Binghamton, NY 13902

Abstract

The creation, behavior, and performance of particle systems in computer graphics has been a balance between control, visual accuracy, and CPU utilization since their inception. A tradeoff must be made between a physically-accurate visual representation and computation time. The most mathematically-correct way to represent particle systems is through the application of Newton's laws of motion. This solution, in which the particles are aware of each other, can be computationally expensive, usually to the order of $\theta(N^2)$, where N is the number of particles in the system.

A new and better way of representing, controlling, and modifying the behavior of particle systems is proposed and demonstrated. Here, in addition to the global environment that determines the behavior of particles everywhere in space, the particles move through a set of localized environments, in which each environment acts only on those particles that are within the confines of a particular environment. With this system, the parameters of each localized environment are adjusted and provide fine control over the behavior of the particles, thereby producing visual results comparable to those that would be obtained from a physically-correct solution in which the particles interact with each other. The proposed method results in a significantly improved performance of $\theta(N \cdot L)$, where L is the number of localized environments. By changing the parameters of the local environments, a high degree of flexibility in the types of phenomena being modeled can be obtained, thereby allowing the same program to simulate a wide variety of physical systems.

Introduction and History

The history of particle systems goes back to 1983 when William T. Reeves published his inaugural paper "Particle Systems: A Technique for Modeling a Class of Fuzzy Objects" (Reeves [1]). In this paper Reeves simulated a virtual collection of particles by applying Newton's basic laws of motion; thereby creating computer graphics elements that exhibited fuzzy properties. This new paradigm enabled the modeling of effects such as snow, rain, fire, clouds, and swarming bees.

Over the years many advances in particle systems have been made. Some of the more notable ones were: the use of stochastic modeling for rendering trees (Reeves [2]); the modeling of fireworks (Tseng-See Loke, et. al. [3]); the use of particles to model the flocking behavior of birds (Craig Reynolds [4]); the use of particle systems with a surface orientation (Richard Szeliski [5]); the use of particle systems with given properties sent through object models to paint, dissect, and modify three-dimensional objects (Alex Pang [6]). Since particles apply a subset of Newton's basic laws of motion, they can be used to create complex visual results, which often mimic the complexity of real objects visible in nature.

Environment and System Interaction

We have noticed that most existing particle systems are limited to modeling one class of phenomena. We propose the creation of an environment that easily allows the user to modify and change the system in ways that will lead to a more controlled and flexible particle system whose behavior and appearance are not limited by a set of initial parameters.

There are two major parts to a particle system: the intrinsic behavior of the particles themselves and the environment. Intrinsic behavior is influenced by such attributes as particle color, velocity, lifetime and movement pattern. The environment consists of a set of external factors which act on the particles; these factors usually have an effect on all the particles in the system.

Our system provides a dynamic environment that has a strong and interactive participation in the modification and adjustment of the particle system; in return, the particle system will also contribute to the modification of the environment. The mutual interactions between the environment and the particles result in an open-ended modeling system that allows the attributes of the system to be easily changed [*fig. 1*]. With this

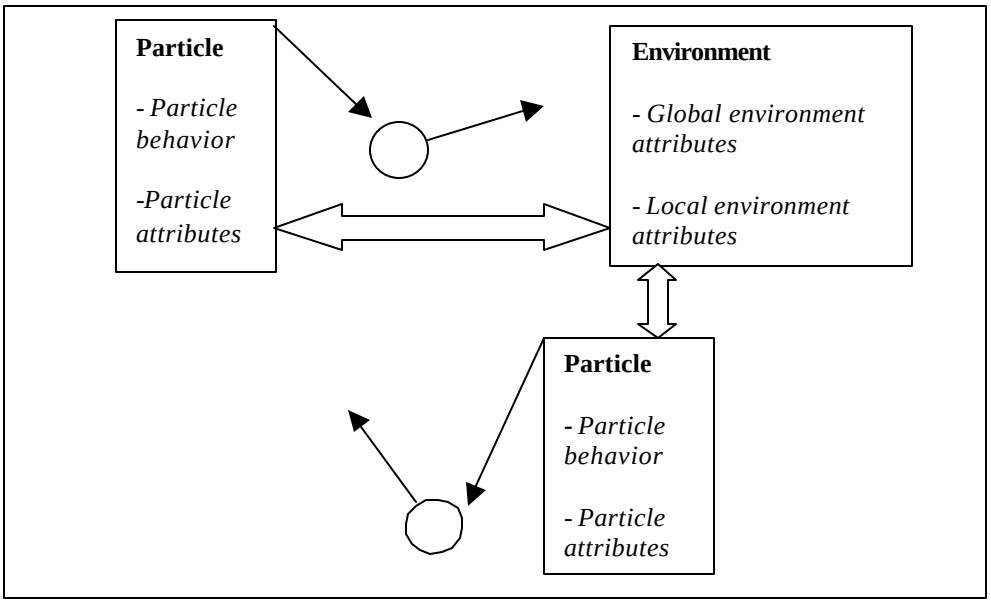

Fig. 1. In this example the particles interact with their environment, modifying the behavior of the environment, which in turn has an influence on the particles themselves.

approach, the particle system is not restricted to modeling a single physical system, as, for example, a waterfall. Instead, more complex and controlled shapes can be created, thereby permitting a wide variety of visual results.

The Localized Environment

In addition to providing for mutual interaction between particles and a global environment, we have added a second more important property: something we call localized environments. When creating a model that includes a particle system, the modeler will usually create only one environment for it. In many cases this is sufficient, since global parameters normally need be applied just once to the whole system. An example is gravity [*fig. 2*]. A single global environment is sufficient to handle any

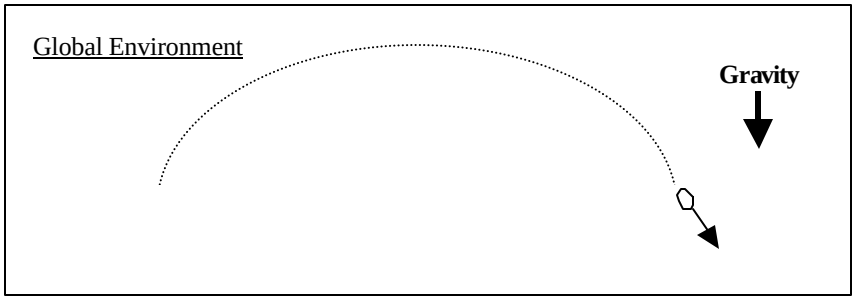

Fig. 2. A global environment where the particle is only affected by one parameter value.

situation in which there is little interaction between the particle systems and the environment; which is usually the case when modeling a simple system that does not dramatically change over time.

For more complicated situations we suggest that the environment be subdivided into smaller more localized environments, in which each environment has its own set of properties. Then the particles will not be fully influenced by the global parameters of the general environment. Instead, their behavior will be dependent on the local environment in which they find themselves [*fig. 3*].

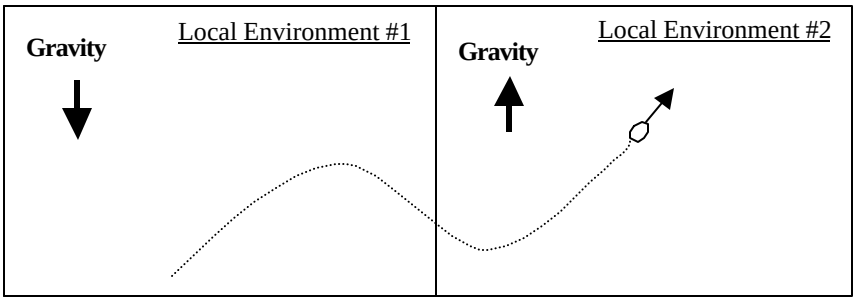

Fig. 3. Two localized environments influencing a particle in two different ways.

As the environment is broken into smaller pieces, each piece will be able to exert a local influence on the particle system. As each particle travels through the global environment as a whole, it will end up passing through the boundaries of different local environments. Each of these local environments will then be able to influence a particle with a different set of parameters. The result is that a more complex pattern of behavior will be exhibited as each particle's path and properties are adjusted according to which environment it is traveling through.

A local environment can have several attributes that affect a particle's behavior and the resulting visual output of the particle system. The attributes that can be modified in a local environment are dependent upon the characteristics of the particle system. Some of the attributes of a particle system that could be modified are given in Table 1.

Table 1: Possible Properties of Particles in a Particle System

Velocity	Size	Position	Shape
Lifetime	Transparency	Weight	Rotational Constant
Orientation	Color	Texture	Material Property

The fact that specific changes can be applied to certain particles within particular areas in a local system provides an advantage over generalized systems where change is more global and can therefore produce only general results.

The attributes of the local environment are not static and can also vary over time. By changing the attributes of the local environment, the resultant visual display of the particle system is then changed. There are two ways of modifying the local environment: attributes can be changed by a trigger, or continuously (sequentially) over time.

To change the local environment by a trigger, the modeler specifies an event that must occur to cause the change in the local environment. Specifically, the change would

be triggered when some attribute (color, speed, age, etc.) of a particle that has entered the local environment falls within a given range or exceeds a certain threshold. The specified change is then implemented in the local environment [*fig. 4*].

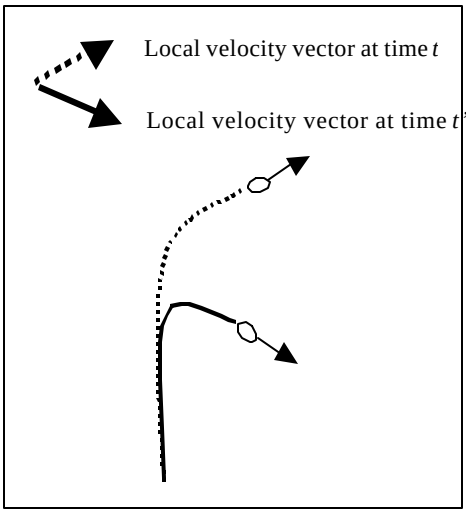

Fig. 4. Change in particle behavior as the local velocity vector changes.

An environmental change may not be necessary at every time step, since this might cause change to occur too rapidly. To prevent this, a rest period for the local environment can be specified. During this time, even if an event triggers a change in the environment, it is not executed until the environment has finished resting the specified amount of time. The local environment can trigger multiple behaviors, depending on the attributes of the particle system triggering it. For example, a particle's color could trigger a velocity change and a particle's weight could trigger a life expectancy change, all within the same local environment.

The second way of changing the attributes of a local environment is to change them at certain points in time. In this case, the modeler specifies the amount that an attribute should change at each moment in time. For example, one can specify that the lifetime of a particle be decremented by a certain amount to a minimum of 0, every 10 time steps. The advantage of this method over having an event trigger a change is that the attributes of the local environment continue to change regardless of which particle has entered its boundaries. In other words, if attributes are changed at specified points in time, there will be a continuous change in these values, no matter what the particle's attribute or position in space is.

Advanced Local Environments

The use of local environments can be a powerful tool for particle systems. Yet there are several enhancements that can be added to the local environments to increase the visual complexity of the particle system while still maintaining a simple interface. When implementing a local environment, one wants to maintain a high level of control

over the system without being forced to deal with redundant and unnecessary variables and modifiers. Some ways of working with these control issues are given below.

Local environments can have different sizes. In areas where there should be a high degree of control the modeler would place more and smaller environments. In other areas where little change in the particles' behavior is required, fewer and larger environments can be employed. As an example, take the modeling of the smoke from a candle. The rising smoke is fairly stable at the top of the flame, but as it rises, its behavior becomes more chaotic. If fixed-sized local environments were to be used, some of them would be too large and consequently would not influence the particles' behavior enough to create any interesting behavior. Other environments would be too small and might cause behavior that is too complex. A better solution to this problem is to model the candle smoke using local environments of different resolutions. The lower part of the model could be filled with larger-sized environments, since the particle movements would be fairly uniform at this level (rising in a straight column from the base). Farther up the column more local environments would be inserted, thus giving the particles in that region a more chaotic type of behavior [*fig. 5*]. Most of the particles will travel

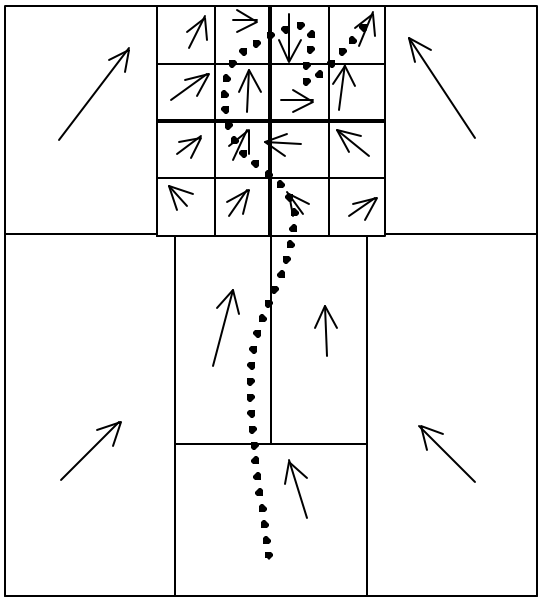

Fig. 5. Modeling smoke, in two dimensions, using environment sizes of different resolutions.

through the coarse environments in a similar way, but, as they rise, the presence of more local environments will change their paths at a finer scale and in more complex ways, thus making the visual behavior more chaotic. Additional realism can be obtained by inserting some random behavior into the system.

Environments can also be set up hierarchically; allowing an environment's children to inherit the properties of its parent. For example, it can be specified that a parent environment change a particle's color component by a certain value. Then, by adding child environments to the parent, the color will be automatically transferred to the child. As a result, an environment will modify a particle's behavior according to its own parameters and those inherited from the parent. The advantage of this is that a general

behavior can be specified at the parent. This behavior will be applied automatically across several environments, while still allowing unique parameter modification in specific child environments.

The local environments do not have to display static behavior. Instead, their attributes could be modifiable over time and space. For example, their positions and sizes can vary over time. This allows the modeler to specify a bounding box for the local environment whose size and position can change. This changing local environment is transported across the global environment, thus allowing certain aggregate characteristics of the particles' behavior to be transported from one region to another as the system evolves.

When specifying local environments, borders can be extended to overlap with other local environments. In other words, environment boundaries do not have to terminate at a border. The advantage of overlapping boundaries is that edge effects can be eliminated. If a drastic change in parameters occurs as a particle moves from one environment to the next, a rapid or discontinuous change in the particles' behavior or visual representation will be perceived. To create a more continuous change in behavior between environments, a certain amount of overlap can be specified. In the overlap area the parameters are averaged, thereby making the perceived changes less severe. For example, figure 12 shows a two-dimensional world with two local environments. Environment One has a local velocity vector with an angle of –45 degrees. Environment Two has a local velocity vector with an angle of +45 degrees. In the middle of the global environment, these two local environments overlap. Within these overlapping regions the two velocity vectors are averaged, with a resultant velocity vector of 90 degrees [*fig. 6*]. As a particle passes between the two regions, its velocity is thus changed in a more gradual way.

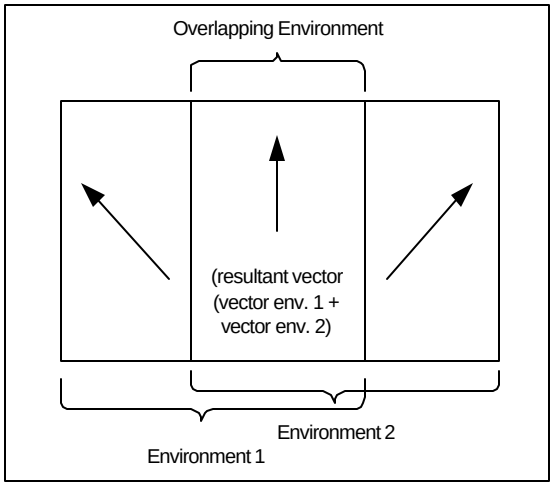

Fig. 6. An example of overlapping environments.

Computation Complexity

One of the major advantages of using localized environments is that the modeler can create more complex particle behavior without a great deal of additional computational effort. The major factors that affect computation time are the total number of particles in the system and the number of local environments. The most important of these is the number of particles. For each frame of an animation, every particle has to be accessed and manipulated at least once. Immediately, this leads to a computation time of $\theta(N)$, where N is the number of particles. In the most simple kind of simulation the particles are not aware of each other; there is no collision avoidance and there are no forces between particles. Any interactions that occur are usually handled heuristically, are experimentally predetermined, or simulated by changes in environmental parameters.

In more complex systems particles are aware of each other and can interact with each other. Particle awareness is needed if the location of a particle relative to all the other particles must be known, as, for example in collision detection, or if the particles are to be represented as point masses that exert forces on each other. At each time slice, the distance between each particle and every other particle must be computed, and from that computation, each particle's behavior is modified. Thus the computational time is $\theta(N^2)$. Particle systems of complexity $\theta(N^2)$ are often used in physics for the modeling of macro/microscopic events such as the behavior of subatomic particles and the evolution of galaxies [7]. The advantage that this true-to-life modeling of physical events offers to physics is a disadvantage in the field of computer graphics because of the high computational cost. For computer graphics, life-like modeling is frequently of secondary importance to quick execution. It is usually preferable to employ heuristic shortcuts instead of mathematically intensive, physically valid computations in order to achieve a result – provided, of course, that there is not a major loss in visual quality. Particle systems in computer graphics are more likely to use heuristics and other tricks to reduce the amount of computation, while still creating lifelike physical effects.

In our system the second major influence on computation time, after number of particles, is the existence of the local environments. The particles in a localized environment are not aware of each other, they are not able to implement collision avoidance, or exercise forces upon each other. On the other hand, the particles are aware of the local environments. As a particle travels through a local environment, a check is made to see if it is still within the bounds of the current environment. So, in a worst-case scenario, a single particle computation time of $\theta(L)$ is required, where L is the number of local environments. Taking into account both factors gives a computation time of $\theta(N \cdot L)$. Thus the localized environments are a compromise between an unaware particle system of complexity $\theta(N)$ and an aware system of complexity $\theta(N^2)$. The particles cannot influence each other directly, but a particle's interaction with a localized environment can influence other particles' behavior through, for example, the use of triggers in the local environment. The result is that one can model complex behavior with particles that are semi-aware of each other while still maintaining a computational complexity closer to $\Theta(N)$, as long as $L \ll N$.

The Particle System Simulator

To test the concept of local environments, a program was written that would allow for the creation and manipulation of particles and particle systems. The program was written for Windows using Microsoft Visual C++ and the OpenGL graphics libraries. The resulting particle system simulator allows the user to test and manipulate various types of particle systems. The modeler can run simulations consisting of multiple-particle systems whose attributes can be changed over time. Particles themselves can be represented as single point sources, bitmaps, or other geometric primitives (spheres, rectangles, etc.). The simulator allows the user to create and manipulate any number of local environments. Various changes can be made to the particles' characteristics, either through triggers in the local environment or through incremental changes over time. An attempt was made to create an open ended environment for the creation of various types of particle effects.

Below is a screen shot of the user interface [*fiq*. 7] to the particle system simulator. It consists of several dialogs that allow the modeler to alter parameters that affect the behavior of a particle system. The bottom "local parameter" dialog modifies the variables of the local environments; triggers and behaviors of a local environment are set here. The dialogs on the right modify global parameters (bottom) and the generator (top), which initializes and creates the particles.

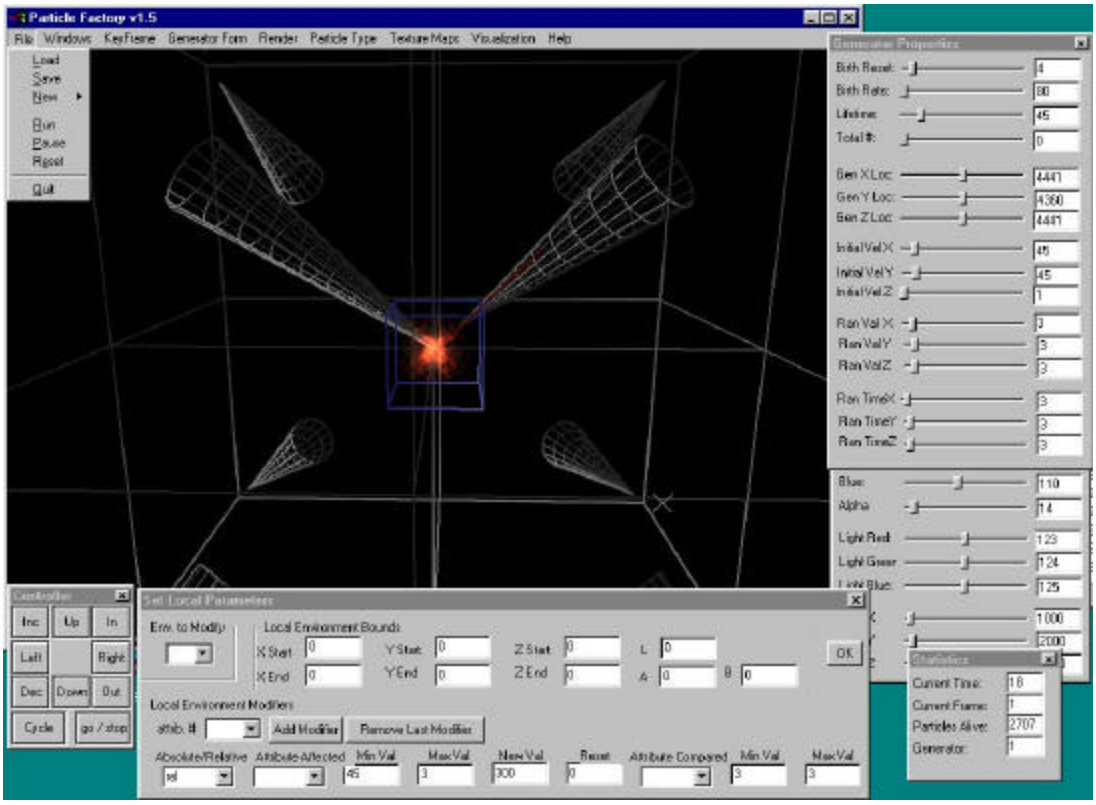

Fig. 7. A screenshot of the Particle System Simulator

Figure 8 below shows a screenshot of an environment created by the Particle System Simulator. One can see five local environments represented as white rectangles. Any particle leaving the generator (yellow rectangle) and moving through these local environments will have its characteristics manipulated by these environments.

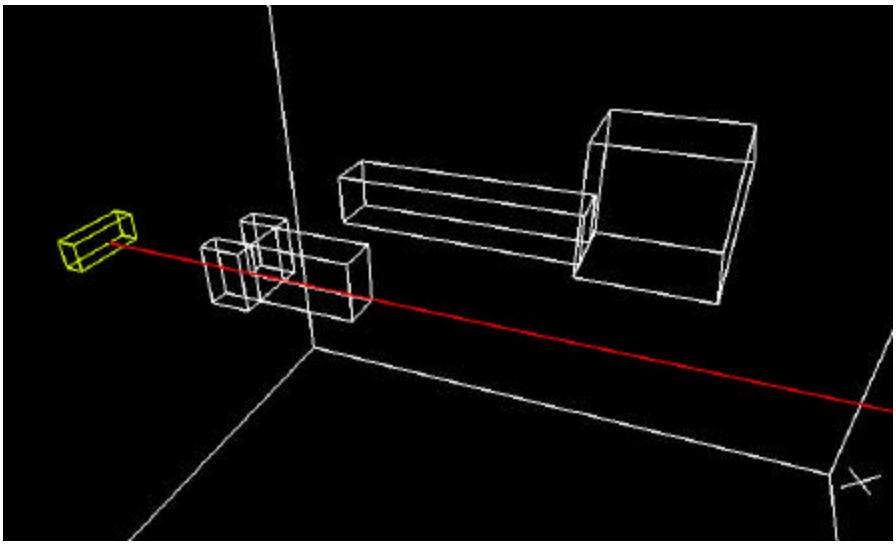

Fig. 8. Simulation with 5 local environments.

Figure 9 shows a sequence of images from an animation of a particle system as it moves through the above environment. In this example a full rendering has been done, and the local environments are not displayed. In looking at the animation one can see that as the particles enter the local environments, their paths and color values are being manipulated.

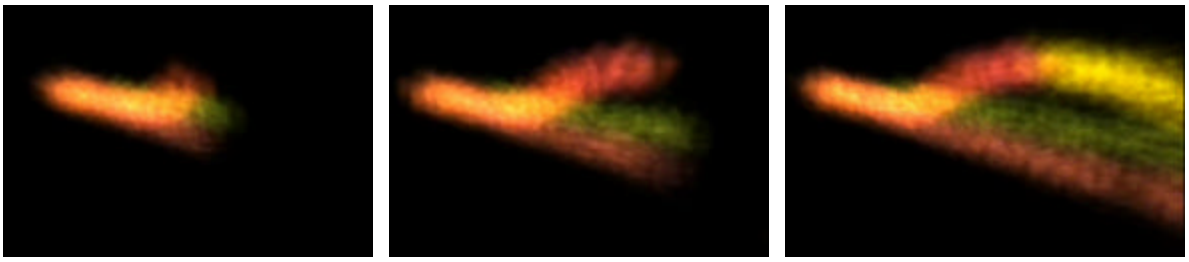

Fig 9. Particle animation rendering with local environments.

Figure 10 shows another sequence of images that demonstrates how two local environments can affect the behavior of a particle system. Environment 1 (top) modifies the *y* velocity vector of the particle in a downward direction. Environment 2 (bottom) modifies the *y* velocity vector of the particle in an upward direction. As a particle gets ejected from the generator (with an initial velocity along the *x* plane), the velocity component is modified by the local environment, resulting in the sequence of images depicted in the figure.

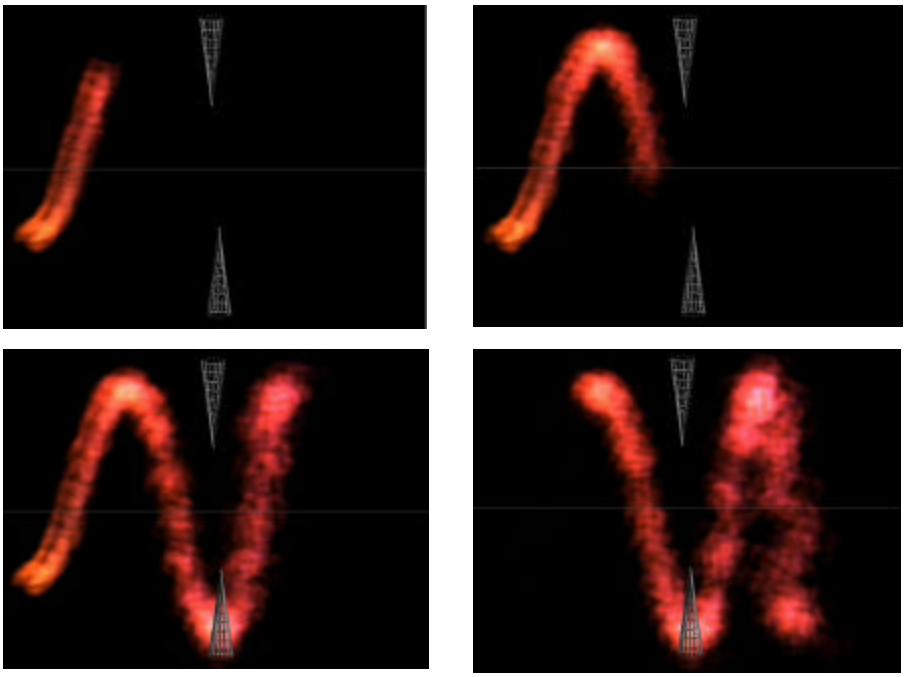

Fig. 10. Two environments modifying the velocity vector of a particle system.

Figure 11 below shows a sequence of images from an animation of a particle system consisting of two generators that produce particles that move through changing local environments. The first image of Figure 11 shows the initial generator for the first particle system (blue). The second image shows the addition of a second generator (green) creating a second set of white particles. The third screenshot shows the addition of four local environments (gray) that adjust the particles' attributes. The aggregate visual effect is that of an explosion.

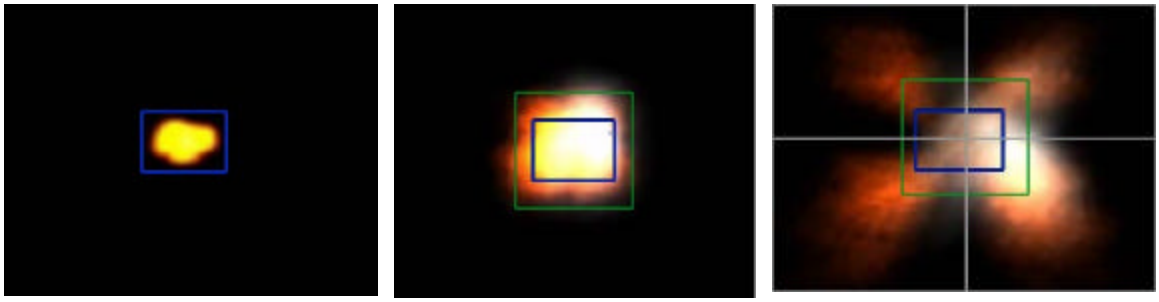

Fig. 11. A sequence of animations showing two particle systems being generated and modified by local environments.

The animations from which these images were taken (as well as other animations produced by the particle system simulator) are available as AVI files that can be downloaded from our particle systems web page [8].

Performance of the Particle System Simulator

Several tests were completed to determine the effects of the number of particles and the number of local environments on the rendering speed of the particle system simulator. Table 5 below shows the results of one set of tests:

Table 5: Rendering Time of a 120 Frame Animation Using Local Environments

# of local env. / # particles	*5,000 particles*	*50,000 particles*
1 local environment	121 seconds	511 seconds
4 local environments	409 seconds	2192 seconds
64 local environments	1735 seconds	6+ hours

From this table it can be seen that increasing the number of environments results in longer rendering times. This is an important factor to consider when deciding how to set up the environment for the particle systems. If one chooses a system with a large number of local environments, a finer level of control exists over the behavior of the particles, but at the expense of longer rendering times. If the number of environments approaches the number of particles, then the rendering time required in the worst case is $\theta(N^2)$, which is equal to the computation time of a particle-to-particle based simulation. Since mathematically-correct physical modeling is not the goal in computer graphics when an approximation will be just as effective, the modeler must choose the smallest possible number of local environments that will yield the desired effect.

Conclusion

By introducing local environments into a particle system simulation, more complex and varied visual behavior can be obtained. A stronger emphasis placed on the environment in which the particle system evolves as well as on the interaction between the particles and this environment can produce complex visual effects without having to artificially limit or predetermine the behavior of the particles. With local environments many different particle effects can be modeled without having to perform any major modifications to the computer code that generates the particles. Rather than having to create a particle system that is specific to each phenomenon being simulated, the desired effects are produced by altering the parameters of the local environments within the particle system.

A particle system is a tradeoff between realism and computational shortcuts. The more fully the laws of physics are obeyed by a particle system, the more processing power is required. If a mathematically-correct particle system is not the main emphasis of the program, shortcuts must be employed to lower the processing time for the creation of particles, while maintaining the principal visual and behavioral representation that the modeler wants. By working with the inherent visual characteristics of a particle system and by applying local environments, the modeler can save computational resources while still creating a rich visual rich result. A particle system with local environments offers a balance between a computationally-expensive particle aware system, and a simple

system, where particles are not aware of each other. This can be an ideal solution for those who wish to model a wide variety of particle behaviors, but do not have the resources available for a full particle-to-particle aware implementation.

References

[1] Reeves, William T., "Particle Systems- A Technique for Modeling a Class of Fuzzy Objects" ACM Transactions on Graphics, Vol. 2, No. 2, April 1983, pg. 91-108.

[2] Reeves, W., Blau, R., "Approximate and Probalistic Algorithms for Shading and Rendering Structured Particle Systems" Proceedings of SIGGRAPH '85 , pg. 313-322, 1985.

[3] Loke, T., et. al. “Rendering Fireworks Displays” IEEE Computer Graphics & Applications, May 1992.

[4] Reynolds, C., "Flocks, Herds, and Schools: A Distributed Behavioral Model" Computer Graphics, Vol. 21, No. 4, July 1987.

[5] Szeliski, R., Tonnesen, D., "Surface Modeling with Oriented Particle Systems" Computer Graphics, Vol. 26., No. 2, July 1992.

[6] Pang, A., Smith, K., "Spray Rendering: Visualization Using Smart Particles" IEEE Computer Graphics and Applications, September 1994.

[7] Lemmens, K., “An Investigation, Implementation, and Comparison of 3 Important Particle Simulation Techniques.” (Mod. February 27, 1998). Web: <http://dutita0.twi.tudelft.nl/DV/Staff/Lemmens/MThesis.TTH/report.html>

[8] Kutz, D., “Particle System Local Environment Resources.” (Mod. September 5, 2000). Web: <http://www.elemeta.com/paper/>