Particle Systems in SFX

Haraldur Örn Haraldsson
Linköping University
Campus Norrköping
harha404@student.liu.se
vfx-tricks of the trade

ABSTRACT

Particle systems have become a standard type of technique to use for special effects in the film industry and also for simulations in the video game industry. Particle systems are used to model certain “fuzzy” things that are dynamic and fluid, i.e. are non-rigid and don´t have well defined surfaces, e.g. fire, smoke, water, explosions, hair, fur or even more abstract effects as glowing trails.
This report will present a broad overview of the area of particle systems and look into a few papers on the subject from it´s early implementation, plus a more updated view of it´s usage in the industry today.

Keywords: Particle Systems, Particle Simulation, Flock, SFX.

1. INTRODUCTION

Particle systems have been widely used in films for the last decade or so doing a lot of effects that we have begun to take for granted, such as the aforementioned explosions.
You don´t have to look far to find effects that are made by (or at least based on) particle systems in films or even computer games. To use particle systems greatly enhances the possibility to visualize random and unpredictable effects such as explosions in a relative easy and fast way.
A few examples include the green cloud of gas in *The Mask* which was made entirely out of millions of tiny 3D particles, the hurricanes and whirlwinds in *Twister* which were made with particle systems and last but not least all the water/waves in *The Perfect Storm* (which takes place almost entirely out in the open sea) were created with a specifically built particle system for that movie.
The earlier systems were always built specifically for each film which suited the need for that specific simulation, e.g. fire, smoke or maybe hair, which made the process quite time consuming.
Today standalone particle systems tools are available for designers to play around with and create effects in an intuitive and straightforward way, although for bigger projects, like the aforementioned *The Perfect Storm*, specific needs always tend to be addressed.
This report will look back at the beginning of particle systems, how it works and some different aspects of a particle model. Finally, how particle systems are used today is explored.

2. BACKGROUND

Already in the 1960s there were video games that used a form of particle systems, 2D pixel clouds to simulate explosions. The first time a dynamic particle system for computer graphics was presented was in 1983 by Reeves[4], where he introduced a method for modeling fuzzy objects, for simulating e.g. fire, explosions and trees.
The paper is written after applying a particle system to the Genesis Demo sequence in the film *Star Trek: The Wrath Of Khan* (Figure 1). He describes in the paper basic motion operations and data representing a particle, which still applies to this day.
A more in-depth look into the particle system model will be explored in the next chapter.

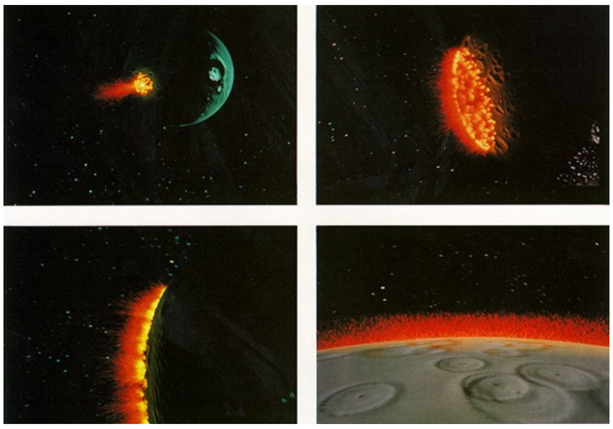
Figure 1: Genesis Sequence from Star Trek: The Wrath Of Khan

Reeves´ particle system built complex objects from sets of simple, volume-filling primitives, but this produced such irregular and complex structure that shading and visible surface calculations were extremely hard to do. In the 1985 paper [3], Reeves introduced methods to address this issue by using approximation and probabilistic algorithms to analyze the performance of the algorithms to understand the costs of their stochastic(enabling to model complex motions with random variation) modeling approach.

This framework was further extended in [6] enabling the solution to use parallel computations, where one virtual processor is assigned to each primitive data element:

- one to each particle,

and during the rendering:

- one to each pixel-sized particle fragment,
- and to each pixel.

Reynolds in the 1987 paper [5] also further extended the system into an area which has a big influence still today, *behavioral systems,* which models complex movement of objects, e.g. flocks of birds. At this point the particles are not just individual points but were objects capable of interacting with each other and simulating complex behavior. Reynolds explored the possibility to make a system of particles, the particles in this case being the birds or herd of land animals, and rather than having to script the individual paths of each bird this particle system would be used to model the behavior of the birds. Each simulated bird would be implemented as an individual actor and react and move according to it´s environment.

Reynold´s method is successfully still used today to model crowd simulations, where the film *Lord of the Rings: Two Towers* is a good example of crowd simulation (the vast army of ogres during battle sequences were simulated a good deal).

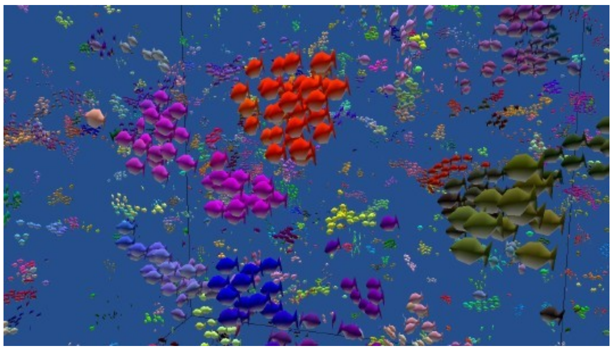
Figure 2: An example of Crowd and Flock simulation

Szeliski and Tonnesen in their paper [7] introduced the idea of orientated particles to display among other things, clothing. Here the particles are interconnected with springs to make up the cloth and then simple dynamic computations are done to simulate the movement.

The advantage of this method is that the computations can be done in real time but those systems are quite limited in their behavior and complexity. Today systems like Kipfer and Segal [2] introduced, use the GPU for computations and are fully capable of rendering millions of particles in real time, even though that the control and interaction over the particles are still limited.

The process of using particle systems have also been made a lot easier in recent times with full fledged tools specifically made for doing these type of simulations, e.g. explosions, crowds etc.

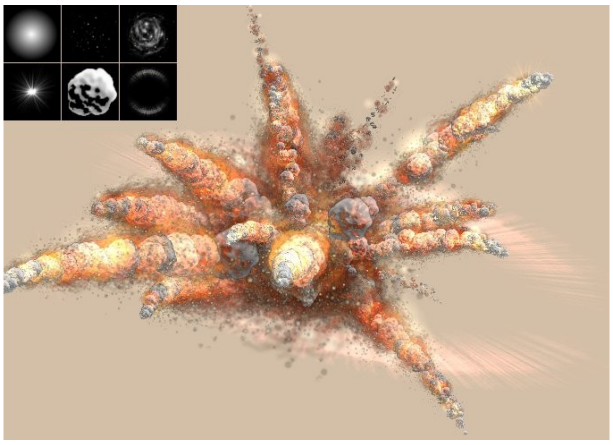
Figure 3: Particle system used to simulate a bomb explosion, created with the tool *particleIllusion*.

3. PS IMPLEMENTATION

In this chapter the model for a particle system is introduced in three different variations, beginning with the basic Reeves model that is the foundation for which the other systems build on.

Firstly the Reeves[4] system consists of an emitter that creates the particles and updates them accordingly.

This emitter handles the task of deleting the particles that have exceeded their lifetime and creating new ones, giving them their initial attributes. Then the current particles are moved and transformed according to the rules set by the system.

The basic model for a particle system is built on the following properties for each of the particles:

1. Position
2. Velocity
3. Colour
4. Lifetime
5. Age
6. Shape
7. Transparency

And each particle goes through three phases:

1. Generation – the particles are generated randomly around each “fuzzy object” and each of the above attributes are given initial values.
2. Dynamics – the attributes of each particle may vary over time, i.e. position, velocity, colour, lifetime, age, shape and transparency.
3. Extinction – each particle has two attributes dealing with length of existence, age and lifetime. Where they may have criteria for terminating prematurely: going out of bounds, hitting the ground or reaching a certain threshold.

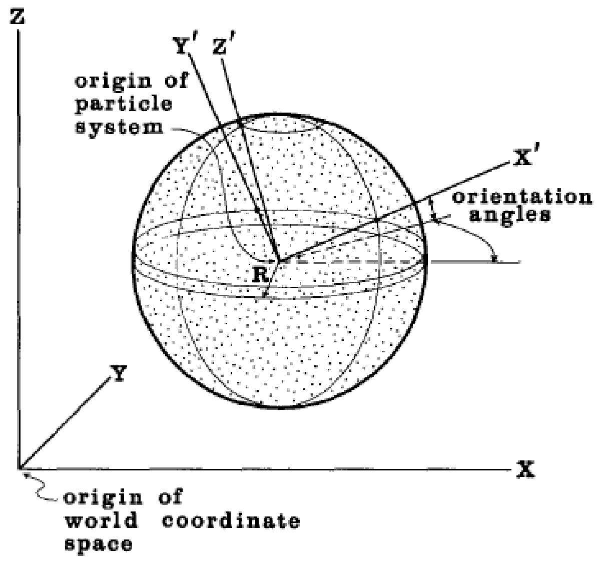

Figure 4: Typical particle system with spherical generation shape.

Particles are generated by using stochastic methods, in one method the designer controls the mean number of particles generated per frame and the variance. Another method generates a certain number of particles per screen area, which is good for controlling the level of detail. The designer can change the number of particles generated as time changes by simple linear functions.

The position, orientation and mass of each particle needs to be known to be able to draw it in a given frame of the simulation. For this the forces that affect the particles (wind, gravity, explosion force etc.) are described, opposed to moving each particle manually. The mass of the particle is known and the force that is applied to it then

$$F=ma \quad (1)$$

gives the acceleration at any point in time.

Derivative of position is velocity:

$$v=dx/dt \quad (2)$$

Derivative of velocity is acceleration:

$$a=dv/dt \quad (3)$$

What is needed next is to know where the particle will be at the time of the next animation frame, as the mass and the forces are known at a particular instant of time. For this, integration is needed, as most of the time the forces are not constant but depend on the velocity or position. In this case numerical integration is needed as an analytical approach is usually too complicated. Some examples of numerical methods suitable would be Euler (first order method), Midpoint (second order) or Runga-Kutta (fourth order). The difference is that with higher order the calculations get more accurate but need more work.

As the system is rendered the particles are represented as tiny primitives, like points, lines or small polygones. More newer versions use textured billboard quad or point sprites, even animated sprites or video-textures can be used. When particles map to the same pixel on screen they are additive, i.e. the color of a pixel is the sum of the color values of all the particles that map to it, which let´s the system to be rendered in order, thusly no hidden surface algorithms are needed. And as the position and velocity are known for each particle it is possible to render them as a streak and achieve motion blur relatively simple.

Particles can also obscure other particles behind them, can be transparent and cast shadows on other particles. They can also interact with each other as mentioned earlier regarding crowd/flock simulation. In Reeves´ work the particle systems did not intersect with other primitives, then the system only needs to handle the particles.

One big advantage of this system is that it has a particle hierarchy, i.e. the particles themselves can be particle systems, where the “child” particle systems can inherit the properties of the “parents”.

Another way Reeves[3] used his system was to model Trees, which he used for the short movie *The Adventures of André and Wally B*. Each tree was created as a particle and the position was also controlled. At this point the rendering method mentioned before, to render the particles in order, was not used. A traditional rendering method was used, but because of the amount of particles in the scene some reasonable solutions were made.

- Self Shadowing – The outside of the tree is more likely to be directly lighted than points inside the tree.
- Ambient Lighting – Based on a similar equation as the self shadowing.
- External Shadowing – Shadows cast from neighboring trees are considered by tracing a line from the top of each neighboring

tree to the light source. Particles above this line are not shadowed but those below have a bigger chance of being in shadow according to the distance from the line.

Each tree has it´s own bounding volume thusly avoiding any intersections, giving the possibility to render each tree independently. This simplifies the process greatly as the amount of particles that have to be loaded simultaneously decreases a lot.

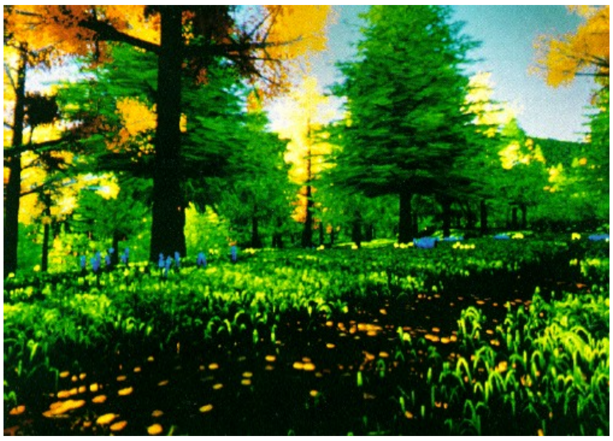
Figure 5: Forest scene from *The Adventures of André and Wally B.*

The last method is the one used by Reynolds [5] when modeling the behavior of a flock of birds. The systems represented was made up of "boids" (bird objects). This system was a "generalization" from the Reeves model or as Reynolds said himself: "a subobject system", where he replaces the dot-like particles with geometrical objects which consist of full local coordinate system and a reference to a geometrical shape and one of the most important aspects being that the individual objects now have orientation. The following is an overview of Reynolds model:

- Each boid is an entire polygonal object.
- Each boid has a local coordinate system.
- Fixed number of boids, they are not created or destroyed.
- Traditional rendering methods are used as there is less amount of boids in the scene.
- Boids behavior is dependent on external and internal state, which means that boids react to other boids around them.

The behavioral model is based on observation of real flocks and flock behavior, and he came up with three primary boid directives:

- Collision avoidance – no collision with other boids or objects.
- Velocity matching – boids attempt to go the same speed and direction as neighboring boids.
- Flock centering – boids stay close to other boids in the flock.

These three needs set the impulses the boids move according to.

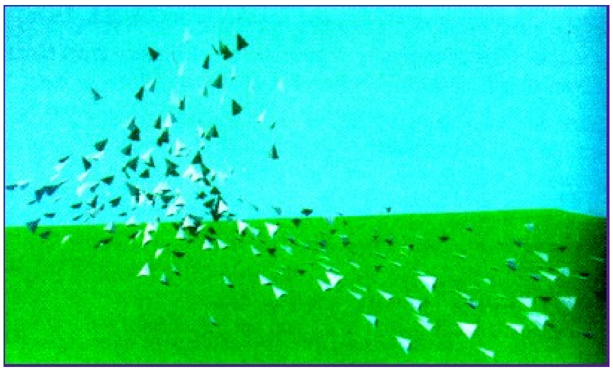
Figure 6: Particle system demostrating a flock of objects as presented in the Reynolds paper.

4. UP TO DATE

As I mentioned before Reeves´s model is one that is still used today as a basic model to extend upon. Here we look at a few more up to date techniques that involve particle systems.

One thing that is interesting to mention is the usage of particle systems in the field of fluid simulations where e.g. Kruger et al.[1] presented a particle system for interactive visualization of a steady 3D flow using millions of particles.

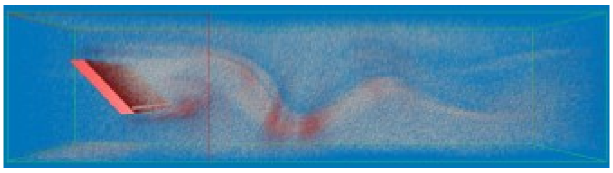
Figure 7: Million particles traced through the flow at 60fps.

This example of course gives thought to rain simulations which also is well suited for particle systems and is usually used for such simulations.

As we have mentioned before it has become important to utilize the GPU for simulations as it considerably speeds up the calculations, and there are systems utilizing that, e.g. when using geometry shaders with particle systems.

Another attribute with particle systems is that they are not limited to be just animated, they can be static as well, i.e. the lifetime of the particles can either be rendered over time or all at once, think the difference between "snow" and "hair". As mentioned earlier the particles are animated, where each particle occupies a single point position in space. But if the entire life cycle of the particles are rendered at once, they will be static, basically strands of material that visualize the particles´ overall trajectory. These strands can be used for hair, fur or grass simulations.

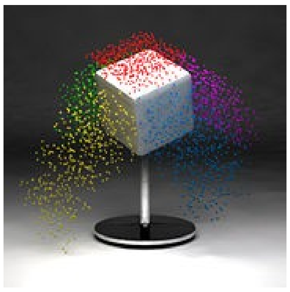
Figure 8: Particle emitter (wiki).

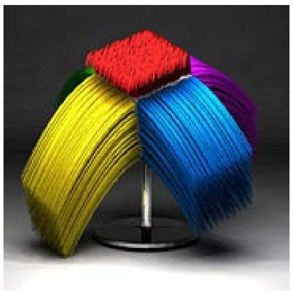
Figure 9: Strands emitter (wiki).

Then there is a wide usage of these type of systems with crowd simulations, like mentioned before, when needing a large crowd to move over a landscape or just to stand still, particle systems give good control to define the actions of the characters that they want to visualize by just using the definitions mentioned before, e.g. the behavioral model.

There are a lot of more examples to mention but one other important thing to mention is that using particle systems today has become quite straightforward, even for the designer. Tools and software have been created that do particle simulations automatically (or almost), thusly making it quite simple to try out different types of simulations, like explosions which is an integrated automatic generated simulation in most tools today.

Examples of particle system tools is the free Particle Systems API[1], and the not so free designer friendly particleIllusion[2] software tool.

5. CONCLUSION

In this report particle systems has been introduced from it´s early inception. Then the implementation of a particle system was presented along with a couple of extended models. Finally the more up to date status of particle systems was discussed to give an overall broad view of the topic of particle systems.

1 http://www.particlesystems.org/
2 http://www.wondertouch.com/

REFERENCES

[1] Jens Kruger, Peter Kipfer, Polina Kondratieva, Rudiger Westermann. A Particle System for Interactive Visualization of 3D Flows. *IEEE transactions on visualization and computer graphics,* vol. 11, no. 6, 11. 2005.

[2] Peter Kipfer, Mark Segal, and Rüdiger Westermann. Uberflow: A GPU-Based Particle Engine. Graphics Hardware, 2004.

[3] W. T. Reeves. Approximate and Probabilistic Algorithms for Shading and Rendering Structured Particle Systems. *Computer Graphics.*, 19(3):313-322 , 1985.

[4] W. T. Reeves. Particle systems – a technique for modeling a class of fuzzy objects. *ACM Trans. Graph.*, 2(2):91–108, 1983.

[5] Craig W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34, 1987.

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

[7] Richard Szeliski and David Tonnesen. Surface Modeling with Oriented Particle Systems. *In Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 185–194. ACM Press, 1992.