Particle Expansion Tool
Nate Usiak

Submitted in Partial Fulfillment of the Requirements
For the Degree of Master of Fine Arts in Visual Effects
at
Savannah College of Art and Design

© August 2013, Nathaniel Paul Hutto Usiak

The author hereby grants SCAD permission to reproduce and to distribute publicly paper and electronic thesis copies of document in whole or in part in any medium now known or hereafter created.

Signature of Author and Date

/___/___
Deborah Fowler (Sign here) (Date here)
Committee Chair

/___/___
Malcolm Kesson (Sign here) (Date here)
Committee Member

/___/___
Patricia Perrone (Sign here) (Date here)
Committee Member
Particle Expansion Tool

A Thesis Submitted to the Faculty of the Visual Effects
in Partial Fulfillment of the Requirements for the
Degree of Master of Fine Arts in Visual Effects
Savannah College of Art and Design

By
Nathaniel Paul Hutto Usiak
Savannah, GA
August 2013
This thesis discusses a tool created to help speed up simulation times for particle based Houdini simulations. A tool such as this would greatly increase productivity by speeding up the effects pipeline. Many challenges were met during the creation of this tool and this thesis outlines all problems that were solved and the issues not yet resolved.
Introduction and Thesis:

The visual effects industry relies heavily on quick turnaround time on shots to make deadlines on commercials, television shows and films. Making these deadlines can require a large amount of computer resources for rendering and simulating. Any opportunity that studios are given to decrease resources needed and speed up work-flow will in turn increase profit and allow for more jobs to be bid. A large portion of time and computing resources in visual effects projects go to particle simulations.

In 1983, William Reyes first introduced particle systems as a means to represent what he calls “fuzzy” objects, or objects that dynamically change their shape over time\(^1\). He used particle systems for the grass in one of the most substantial images in visual effects history, The Road to Point Reyes as seen in Fig.1.1\(^2\). The image was a GL research project that he worked on with Loren Carpenter, Rob Cook, Tom Porter, David Salesin and Alvy Ray Smith, who all went on to found Pixar. Since then particle systems have become very widely used for everything from the “fuzzy” objects Reeves described such as smoke and fire, but also used for other fluid effects. Particle fluid simulations require complex calculations for collision detection and other calculations to attain realistic fluid motion and quality.

Particle separation defines how closely each particle can be to the nearest neighbor particle. The smaller the separation, the more particles there are in a simulation and the higher the resolution. Resolution is used to tell the scale of the simulation, such as a small scale like a glass of water or a very large scale like an ocean or massive river. For larger simulations requiring higher detail, a studio would need to devote a great deal of resources which may not

---

\(^1\) Reeves, "Particle Systems---a Technique For Modeling A Class Of Fuzzy Objects,” 359-359.
\(^2\) Smith, "Alvy Ray Smith pt reyes."
be available to run these simulations. The visual effects industry will benefit greatly from a tool
that will speed up particle based simulation times through the addition of detail by increasing
particle count to require less computational resources than the original simulation would at that
level of detail.

Fig.1.1 The Road to Point Reyes, Created at Lucasfilm Ltd.

Individual particles in a simulation can have many different attributes associated with
them. These attributes store information used for calculations that take place during simulation
and render time. Two commonly required attributes are:

- Position – “P” – containing position information
- Velocity – “v” – velocity of a particle
Position and velocity provide the data to compute the particle at the next frame. In addition, particle ID is an individual number that is given to each point. This unique number can be used as a seed value in functions like random number calls. Using a number unique to each individual particle as a seed value will return random numbers that are unique to that spawned particle. If the particles are to be surfaced with a particle fluid surface node then the particles must have mass and pscale. These are used by the fluid surface node when the surface calculations take place.

Many studios use tools that will cut down on simulation times to increase resolution of particle simulations. Marcelo M. Maes, Shuntaro Furukawa, Daniel P. Ferreira and Jun Saito outline the use of a particle expansion technique presented at Siggraph 2009. This technique uses an addition of points at render time to reduce disk space for cached simulations. Points are spawned around seed particles in a sphere. To break up clumping, a three dimensional noise look-up is used based on the particle ID. This method produced desired results using ninety eight percent less particles, decreasing the required disk space for the simulation to forty five times smaller\(^3\).

**Explanation of Methods:**

Many different methods can be used to expand particle simulations. Some methods can be faster than others but do not produce results as detailed as other methods that may take longer to process. Each method requires certain data to be attached to the particles that are used as the seed particles and each method uses different mathematical algorithms.

The creation of a tool of this nature requires the use of trigonometric properties. For

\(^3\) Maes, "Multi-point expansion at render time," 1-1.
spawning particles around a seed particle in a spherical arrangement the algorithm relies on several pieces of data being present.

- Seed particles position and particle ID
- Radius of the sphere in which the new particles need to be spawned
- Number of particles to spawn

A random number is generated using both the seed particle's ID and an ID number of the spawned particle. This will allow each spawned particle to have a different random position. Each group of spawned particles will be in different arrangements around each seed particle. The radius is multiplied by a random number generated between zero and one. For example in X an equation such as this is used:

\[ X = ( \text{rand}(\text{seed}) \times 2) - 1 \]

The function uses a seed value for the random number generated. This results in a number that ranges from negative one to positive one. The resulting x, y, z coordinate vector is then normalized and scaled by the radius value previously calculated. This creates the sphere of particles around the seed particle.

There are two main particle expansion methods that were pursued for the purpose of this tool. The first and the basis of the second is basic clustering. In the clustering method, points are spawned in a spherical cluster around the seed particles. These particles then inherit the velocity of their parent seed particle and travel along with them in a group as seen in Fig.2.1.
The second type uses a weighted velocity transfer method. The points are spawned around the seed particle in a sphere but then on each frame preceding their creation frame, their velocities are calculated by taking nearest neighbor velocities and computing a new velocity, as illustrated in Fig.2.2.
Issues arose with the velocity transfer method that led to the creation of a different method called the secondary motion method. The spawned particles are given motion where they move closer and farther from the seed particle. This motion is achieved by multiplying the scaled normalized point vector representing the spawned point’s position by a cosine value using the spawned particles ID and the frame number. A third value, an artist controlled parameter, is used in this cosine function as well to scale the results.

**Purpose of the Tool:**

The goal of this tool is to be able to take the cached out particles from a simulation and add more particles to increase the resolution of the simulation and maintain the same look and feel as the original. The basic clustering method of particle expansion will not be able to give the appropriate amount of detail, because it will look more like clusters of points following another point. The weighted velocity transfer method will give a more fluid effect but will be much more resource intensive than the basic clustering method. A middle ground between these two methods is the secondary motion method. All of these techniques have been pursued in the creation of this tool.

**Challenges Involved:**

Initial algorithm testing was written in Python, and implemented in Houdini using an inline Python Surface Operator node or PythonSOP for short. Python is an interpreted language and C++ is a compiled language. This means Python code is basically processed and rewritten to more usable code by Houdini. C++ however is compiled, which means Houdini does not have to translate any of the code. Since Python has to be interpreted this can make it slower than C++ especially for situations that require a great deal of calculations, however it works well for
prototyping. Refinement of the algorithms was performed using Python due to its ease of use for prototyping. After promising results surfaced from the Python tool the process of bridging the Python algorithms to C++ code began. One challenge involved in the bridging of this tool from Python to C++ was learning HDK which is Houdini’s development kit language based on C/C++. There is very little in the way of documentation so a large portion of time was spent breaking apart sample code, rather than starting from scratch.

**Basic Point Clustering Method:**

The first step was to figure out how point creation and parameter interface were handled. A node was written that created a single point that had a position which was controllable by a vector parameter input. Next more point clusters were added around the point defined by the vector position parameter. Then the points need to cluster around input particles which required querying each input particle's position. The input particle's positions became the points the spawned particles cluster around. The input particle system that was used to get the querying position working had no newly born or dying particles. On the first frame input particles are cycled through and the new particles are spawned around each of them.

Parameters for:

- number of particles to spawn
- distance from seed - radius from seed in which they can spawn

On each frame the spawned particles were moved based on their seed particle's new position. The weighted velocity transfer method requires querying each particle's velocity attribute. The velocity attribute of the seed particle is assigned to the spawned particles. Rather than moving each particle based on its seed particle's position each frame, they are moved based on their
velocity. After the velocity movement was achieved, the next requirement was to be able to handle newly spawned particles. If the spawned particle’s seed particle was no longer present, it would kill the spawned particles associated with that seed particle. This allowed for seed particles to die.

**Velocity Transfer Method:**

For the velocity transfer method to work the seed points needed to be put into a data structure that would allow for each spawned particle to query nearest neighbors to get their velocity attributes. The Python velocity transfer tool used a KD tree. A KD tree takes a three dimensional representation of all of the points and, when queried with a position in space, will return the nearest neighboring particles.

HDK has a built in data type called GEO_Pointtree which uses a KD tree architecture to query nearest points and returns the entire object of the nearest particles. On the velocity transfer tool a parameter was created to select how many nearest neighbors to use for velocity influence. These velocities are weighted based on their distance from the spawned point. These weighted velocities are then combined, assigned to the spawned particle, and then used to calculate the particles new position. This method was found to be slow in comparison to just upping the particle count on the original simulation. This led to the creation of the secondary motion approach.

**Secondary Motion Method:**

The secondary motion method is one in which the particles pulsate around the moving seed particles. This method used the clustering method code as its base. The spawned particles follow along the seed particles by querying their position and moving the spawned particles
around the new position. The tool used two parameters to control this movement, “Scale Time for Noise” and “Scale ID for Noise.” Scale Time for Noise scales the time which slows or speeds up how fast they move over time. Scale ID for Noise controls how staggered the particles move in and out in relation to each other. The combination of both of these parameters allows the artist to manipulate the final look of the new simulation.

**Results of Testing:**

Tests were performed using the tool for both dust effect particle simulations as well as fluid particle simulations. The results of the weighted velocity transfer method were not as promising as the results of the secondary motion method.

The Python velocity transfer tool was able to produce decent results as seen in Fig. 3.1, but was found to be very slow. With the HDK, only small velocity transfer simulations were possible.

![Fig. 3.1 Python Velocity Transfer Method Results](image)

*Fig. 3.1 Python Velocity Transfer Method Results*
Segmentation faults occurred when:

- large amounts of seed particles were fed into it
- large amounts of particles were spawned
- large amounts of seed particles died off in the simulation

Two separate dust tests were performed using the secondary motion method HDK tool on an advecting particle simulation; a Pyro advected particle simulation as seen in Fig. 3.2, and a tornado particle simulation as seen in Fig. 3.3. The base for these simulations uses particle advection which is where particles are distributed using the velocity field of a Pyro simulation to drive them. The Pyro advected particle simulation test was able to achieve promising results with the secondary motion method.
A comparative scene was used for testing to see the differences in computation time between the secondary motion method and simply increasing the number of particles in the simulation. The test was run on a particle advection on a Pyro smoke simulation. The scene shown in Fig. 3.4 was run for 120 frames, three times to test the computational time differences.

- Seed particles – ten thousand particles per second – four minutes and thirty seconds
- Expanding one hundred particles per seed using the tool – one million particles per second - five minutes and thirty seconds
- Increasing scale on original particle advection simulation – one million particles per second – five minutes and thirty seconds

This shows that the tool is currently not as fast as increasing the original simulation resolution. More work is needed to optimize the tool as well as altering the algorithm to be able to more
closely match the original simulations motion. The reason that the results look so fluffy is because the particles are clustered around the seed particles and even though they do have motion it is not influenced by their velocity or velocities of other particles around them.

Fig. 3.4 Secondary Motion Compared Results

**Conclusion:**

The visual effects industry will benefit greatly from a tool that will speed up particle based simulation times through the addition of detail by increasing particle count in such a way that requires less computational resources than the original simulation would at that level of detail. Prototyping in Python showed that the weighted velocity transfer method can produce desired results. Through the creation of this particle expansion tool many problems were encountered when rewriting the code for HDK from Python. Some of the problems were overcome but a few remain. Through more experimentation and exploration into HDK's underlying data structures more improvements can be made to speed up the tool. In addition to
speeding up calculations the algorithm used for the secondary motion method can be altered to create more realistic motion. The tool shows promise as a helpful tool that will speed up particle simulation pipelines after such improvements.

**Problems and Weaknesses of the Tool:**

- tool is not multi-threaded
- weighted velocity transfer implementation crashes on larger amounts of particles
- secondary motion method results do not produce realistic motion with fluid simulations

**Future improvements:**

- implement multi-threading
- use GA_Offset rather than GEO_PrimParticle and GEO_Points – GA_Offset is the new data structure designed to work best in HDK
- alter the secondary motion method to scale the radius that the particles are spawned by the distance to the nearest neighbor
- stretch the sphere that the particles are spawned inside along the velocity vector so that the spheres of particles are stretched in the direction of movement
- add noise to the particles movement
Tutorial:

The tool can be used on either a FLIP fluid particle simulation or a standard particle simulation from a POPnet. Each of these will require different settings to be changed for the tool to work correctly. For the tool to work with a standard particle simulation or POPnet the node needs to be placed after either a file SOP with the cached out particles or the POPnet itself. The POPnet should supply all that is needed but in order to render the points a point SOP needs to be added to adjust the pscale attribute to render the particles. A mass attribute will need to be added to the particles if they are expanded from a particle system without mass and are in the end going to be surfaced with a particle fluid surface node.

FLIP simulations require a few more settings to be altered on the FLIP solver node in the DOPnet. Under the Reseeding tab/Particle Motion tab, turn off the Reseed Particles option and check on Add ID attribute, as seen in Fig. 5.1. These will ensure that the particles from the simulation will have the required ID attribute for the tool to work properly.

![Fig. 5.1 FLIP simulation setup](image)
There are five parameters for user control as seen in Fig. 5.2:

- reset frame - should usually be left at one but allows for the artist to change the frame to reset the SOP
- spawn radius - the radius of the sphere that the points are spawned in
- number of points - how many points to spawn around each seed particle
- scale time for noise - seed scaling value that slows or speeds up how fast particles move
- scale point id for noise - seed scaling value that staggers the particles motion in and out in relation to each other

Fig. 5.2 Particle Expansion Tool UI
Bibliography:

   http://www.hosok2.com/project/dataExpansion_2/dataExpansion_2.html (accessed 
   May 23, 2013).

   August 18, 2013).

Maes, Marcelo M., Shuntaro Furukawa, Daniel P. Ferreira, and Jun Saito. "Multi-point expansion 

Reeves, William T.. "Particle Systems---a Technique For Modeling A Class Of Fuzzy Objects." ACM 