Controlled Discrete Streams

—

The Abstraction of Particle Emission

Georg Duemlein

The Industrial Parametrisation of the World
http://www.preset.de/
2008

Abstract

This document explains concepts and creation of procedural particle simulation in Houdini.
The examples demonstrate how the emission of particles can be controlled by using expressions or control objects.

Controlling the emission of particles is the fundamental tasks of any particle simulation. While one can visualise situations where there is a continuous – uncontrolled – particle emission, in most cases the emission is likely to be controlled by an artist. Therefore we need to supply tools and techniques that enable us to do so in an abstract – user friendly – way.

Particle Emission Types

Houdini's particle emitters offer two kinds of emission: Impulse and Constant activation.

Impulse activation births an specified amount of particles every cook of the operator.
Constant activation births a continuous stream defined in particles per second.

As impulse activation births particles per cook it is highly depended on frame rate and oversampling settings. While changing the frame rate during a project is not that common, oversampling a POPnetwork is often used to raise the accuracy of a particle simulation. This is usually done by defining a render take for the particular POPnetwork and results in a differing amounts of particles during render time.

Fig. 1: Impulse activated POPnetwork: 1 and 5 times oversampled

Constant activation emits a defined number of particles per second. It is therefore independent of frame rate and oversampling settings. Though there is a slight difference when oversampled values the look doesn't change in such a noticeable way.

We also need decide carefully if we use $F (integer frames) or $FF (floating point frames) in expressions while designing a particle simulation. If the setup should reproduce a distinctive look it is inevitable to check the influence of oversampling the particles.

Fig. 2: Constant activated POPnetwork: 1 and 5 times oversampled

For the rest of this document we will use constant activated particle sources. The concepts explained can be applied to impulse sources as well and there are cases you might want to combine both types.

Continuity vs. Discontinuity

We can modify continuity of a particle stream in different ways:

Switching it on and off by changing the activation parameter
Changing the birth rate of the stream
Controlling the probability of birth

The Activation Parameter

We can use the modulo function to periodically emit particles: `$F % 5 == 0`
The modulo operation returns the reminder of a division of two numbers - which is a periodic series of numbers between 0 and the divisor. Writing this expression into the constant activation parameter of a sourcePOP particles will turn it into a discrete emitter – emitting a particle every 5th frame.

Fig. 3: A continuous and a periodic particle stream.

The activation parameter is a boolean value. It is activated on each frame the expression results in 1 – which is another form for the boolean value 'true'.

Fig. 4: Four different tests for the the expression $F % 5

The modulo expression produces distinctive periodic patterns. Another way to produce discrete streams is the use of rand(). This function returns pseudo random numbers that can be used to modulate the emission behaviour: `rand($F) < 0.25`
This will cause the emitter to be activated in about 25% of all frames.

Fig. 5: Random Activation: rand($F) < 1, rand($F) < 0.75, rand($F) < 0.5 and rand($F) < .25

We can combine both functions to produce random regularity or periodic randomness. Their true nature will be only revealed at a larger scale.

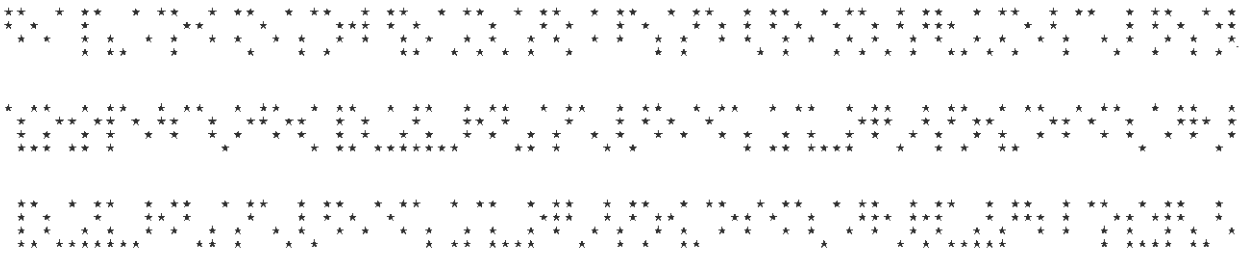

Fig. 6: About 300 frames of four different modulo random combinations

The Birth Rate

The second way to break the continuity of emission is to change the birth rate of the emitter.
The following examples compares an emitter emitting one particle per frame and two other emitters, each reducing the rate by half:

*Fig. 7: Continuous sourcePOPs emitting $FPS, $FPS * .5 and $FPS * .25 particles per second*

Again we can modify the birth rate using the modulo function, which results in a modulated stream of particles. The frequency is determined by the values used.

Fig. 8: Continuous sourcePOPs emitting $FPS, $F % 12 and $F % $FPS particles per second

The random function produces a non-regular number of particles per second. Using $F as seed prevents repetition, while a periodic seed like `$F % $FPS` generates patterns.

*Fig. 9: Continuous sourcePOPs emitting $FPS, rand($F) * $FPS and rand($F % 2) * $FPS particles per second*

Combining Activation and Birth Rate

Activation and birth rate can be combined to generate more complex emission patterns.
In this example the emitter we use the modulo function in the activation parameter:

```
$F % 3 == 1
```

The random function to controls the birth rate by querying the point count of the emitter geometry:

```
rand($F) * npoints("../../null to pops") * $FPS
```

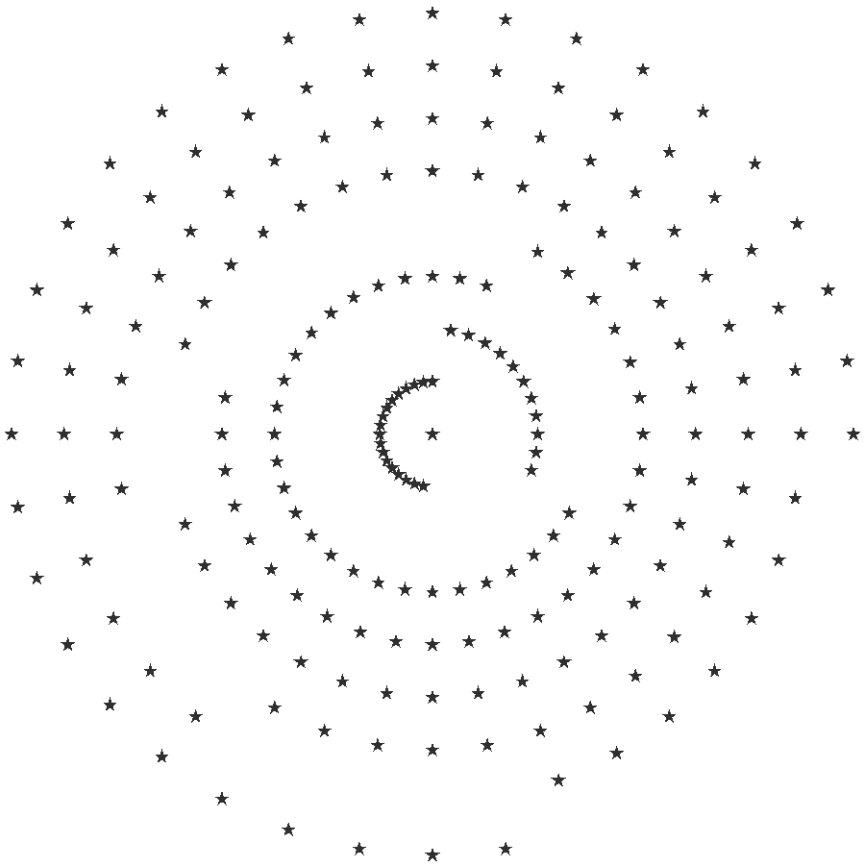

Fig. 10: A continuous activated sourcePOP emitting a random number of points every 3rd frame

Birth Probability

The birth probability parameter describes the likeliness a particle is born at a specific point.
To explore this parameter we use grid emitting a particle per primitive and frame.
The default birth probability of [1, 1] ensures that every particle is emitted.

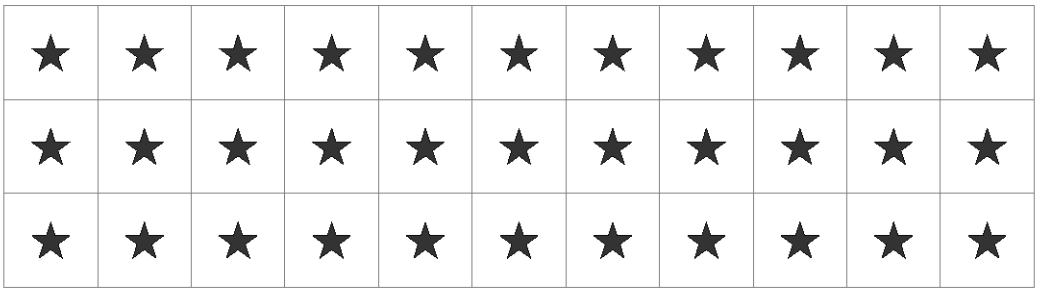

Fig. 11: A gridSOP emitting particles with a probability of [1, 1]

Changing the probability to [.5, .5] is the equivalent of a fifty-fifty chance for emitting a particle for this primitive.

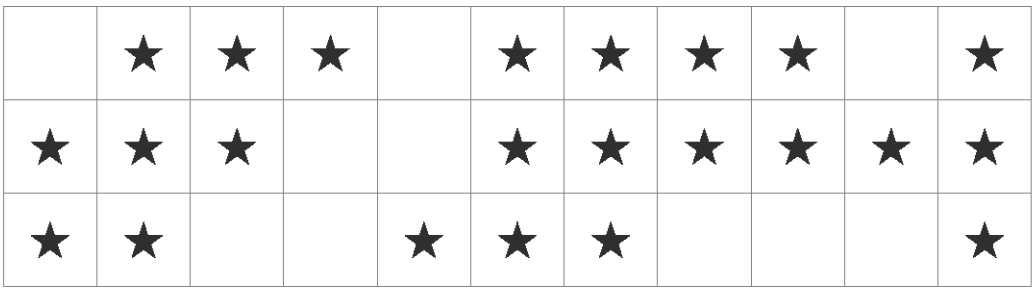

Fig. 12: A gridSOP emitting particles with a probability of [.5, .5]

Different values for minimum and maximum probability assign a random probability from the chosen range.

The parameter is commonly used in conjunction with point attributes like colour ($CR, $CG, $CB) or alpha ($CA).

A potential application is the use of colours to emit particles in a specific shape.
As an example we group points of a grid using a bounding geometry. If they are inside an extruded fontSOP colour them white. Points not in this group will be coloured black.
Using $CR as birth probability emits a particle arrangement in the shape of the font used.

The parameter can be used for effects both interesting and complex. Among the usage just described it can contribute to the variation of a particle stream by adding another layer of variation.

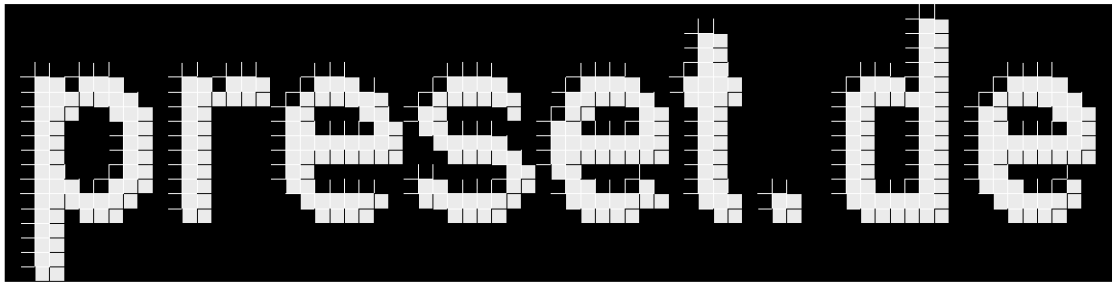

Fig. 14: A gridSOP with coloured points used as base for Fig. 15

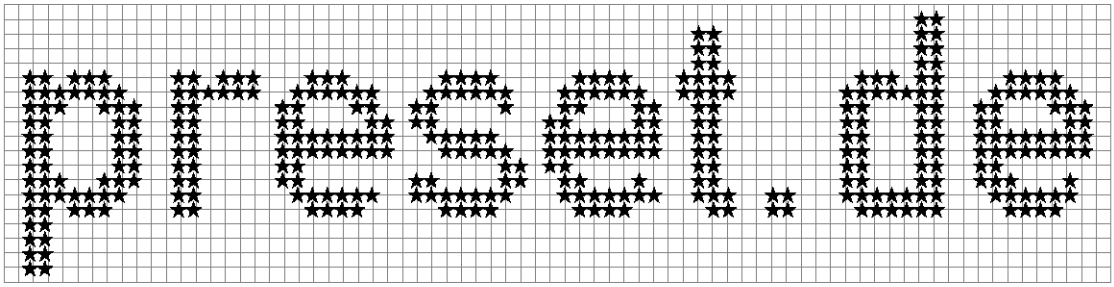

Fig. 15: Particles emitted based on point colours

Controlling the Discontinuity

Combining the techniques descibed it is easy to create some interesting effects.
This is a moving emitter shooting a random amount of particles in regular intervals.

Fig. 16: A moving emitter shooting off a random count of particles.

It is often hard to find the expression or combination of expressions to model a real world behaviour.

A system like a welding robot placing welding spots onto a surface can be adjusted quite easily. But controlling this behaviour using expressions is rather inflexible to changes other than pure timing issues. Also it can become quite complex if the robot needs to interact with others of its kind or the environment.

Setups like the sparkles generated by a car scrapping along a rock wall can hardly described by expressions and doing so is rather complicated if not unmanageable from an artist's point of view.

In such cases it is recommended to use a meta layer. We describe the model in an abstract way using control objects and information from other parts of the scene to modulate the emitting behaviour of particle sources.

This enables the artist to create complex effects without the need of constructing obscure expressions.
The setup in turn is more accessible and can easily be adopted to changed or new situations.

Emit by Distance

As an example of how to approach this task we create a setup that emits particle based on the distance of a control object to a surface.

To do so we create and collect data in the SOP context to be stored in attributes. This information will be queried by the emitter to render the designed behaviour.

Starting point for this exercise is a modulated grid and a probe hovering above it.
The grid emits an random amount of particles while the probe is inside a defined distance to the surface.

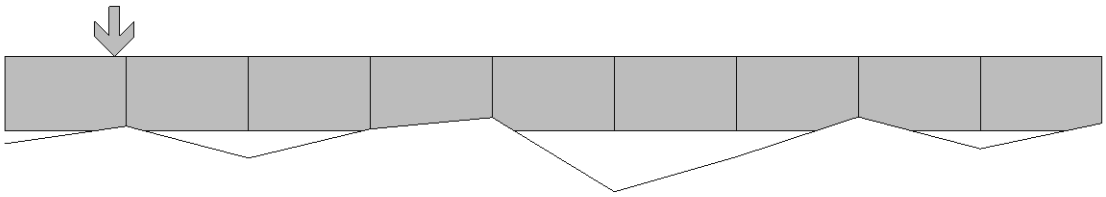

Fig. 17: Schematic view of the behaviour. The gray area indicates the the threshold of emission.

We are using the raySOP to calculate the distance between the probe and the grid.
A lineSOP with two points will serve as template: the probe geometry will be copied to the first point.
The second will be used to calculate the distance to the surface and as template for the emitter.

The raySOP requires a ray. Adding normals [0, -1, 0] to our template indicates the direction. The raySOP moves the emitter template to the surface. We can use the normal of the intersection point to emit the particles perpendicular to the surface.

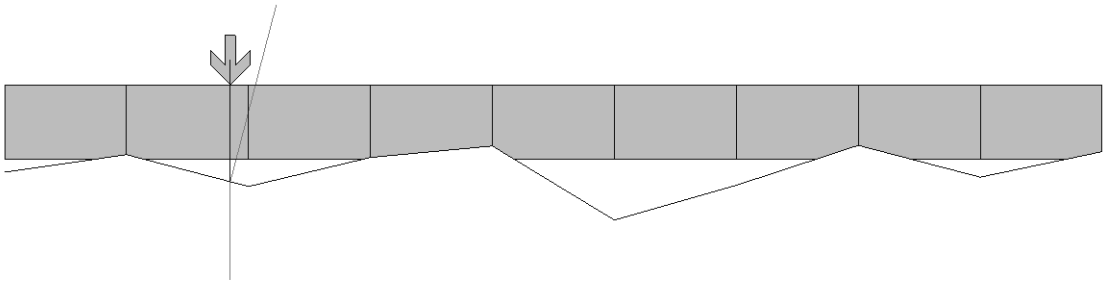

Fig. 18: The raySOP moves the emitter template to the surface and returns the normal of this point.

The distance from probe to surface is stored in a point attribute of type float named *dist*. We use an if expression to convert it into a detail of type boolean:

```
if($DIST < 0.87, 1, 0)
```

We use an detail as the 'emit' is a property of the system and doesn't need to be stored per point.
The threshold value is unique for each system and can differ from the value shown here.
As a visual feedback 'emit' can be used to colour the threshold geometry.

Next step is to pipe the emitter point into a POPnetwork.
The sourcePOP is querying 'emit' using the following expression:

```
detail("../../null_to_pops", "emit", 0)
```

This expression switches the constant activation parameter to 1 every time the probe is close enough to the surface.

To get a variation of the number of particles emitted we write a random function into the constant birth rate parameter. The following one will emit between 25 and 75 particle per second:

```
fit01(rand($F), 25, 75)
```

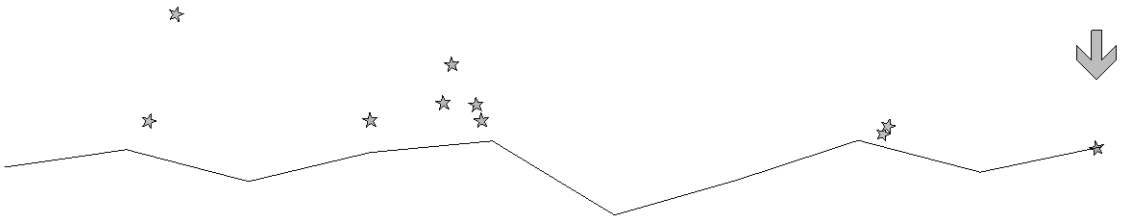

Fig. 19: The completed setup: emitting particles based on distance of a probe to a surface

Depending on the actual use it might be necessary to copy a emitter geometry to the template point.
The setup can be refined by using even more information from the scene and refining the behaviour.

Conclusion

The described concepts and techniques are examples how to design a controlled discrete emission of particles.

Modulo (%) and random (rand()) can be combined with other functions like sinus (sin()) or curves generated using CHOPnetworks or spare channels to design a unique emission behaviour.

Using attributes to model an abstraction of complex setups is a convenient approach when it comes to designing user friendly tools. Artists equipped with such a HDA just needs to connect the required nodes to create convincing effects. In this case study it might be a HDA accepting a surface and a probe-emitter-node. The controls would be 'Maximum Distance' and a "Birth Rate Range" as well as a Seed.

Georg Duemlein
February 2008

References

Wikipedia, Modulo Operation
http://en.wikipedia.org/wiki/Modulo_operation [2/22/2008]