# Liquid Splash | Mist

This page provides information on the Splash/Mist rollout.

## Overview

The Splash/Mist rollout affects the simulation of **Splash and Mist particles**, and enables you to set the conditions for when Splash and Mist should be born automatically. The simulation of Splash and Mist particles can be used for creating effects like waterfalls, rivers, ocean waves and so forth.

Note that the Phoenix Liquid Simulator can simulate different types of particles, including **Liquid particles**, as well as **Secondary Particles** such as **Foam**, **Splash**, **Mist**, and **WetMap** particles. These Secondary Particles exist so that you can achieve a variety of different liquid scenarios. In addition, Phoenix enables you to choose which particles to simulate, depending on your needs.

Because Splash and Mist systems are closely related and share many properties, their parameters are both controlled from within this rollout. When the Splash/Mist rollout is enabled, the Liquid, Splash and Mist systems can be seen as a single system, with adaptive levels of detail.

The Splash and Mist particles adhere to rules determined by the parameter values in this rollout: Liquid can be converted into Splash, Splash can be converted into Mist or back into Liquid, and Mist can be converted into Liquid.

Each particle type has its own life cycle, with rules for when and how they are born and die off. You can learn more about the **life cycle of Splash and Mist particles**, or how they are created or destroyed, by visiting the FLIP Particles Life Cycle docs page.

The Splash/Mist rollout is separate from the liquid simulation, in order to optimize memory and CPU usage. While a normal liquid simulation can produce splash, and if you increase the resolution to extreme values it can produce mist as well, such a simulation would run slowly and consume an incredible amount of resources.

In addition to saving on resources and rendering time, there is also another major advantage to controlling splash and mist separately from the liquid itself: different shaders can be used for shading the Splash, Mist, and Liquid particles. To **render Splash and Mist particles**, you'll need to use the Phoenix Particle Shader. The Particle Shader offers multiple render modes, which enable you to create various fine-tuned appearances for particles such as Foam, Splash and Mist, to achieve realistic looking effects.

For example, to render **Splash particles**, you can choose from the Particle Shader's various **bubble mode** options, such as the **Splashes mode**, so that the particles look like Splashes when rendered. Meanwhile, the **Points** or **Fog mode** can be a useful option for shading **Mist particles**.
Note that Splash and Mist can be simulated in two different ways: either as part of a liquid simulation, by enabling the Splash/Mist Rollout (which is a physical mode), or they can also be simulated as fully separate systems by using a Source to emit Splash or Mist particles (which is a non-physical mode).
UI Path: ||Select Liquid Simulator object|| **> Modify panel > Splash/Mist rollout**

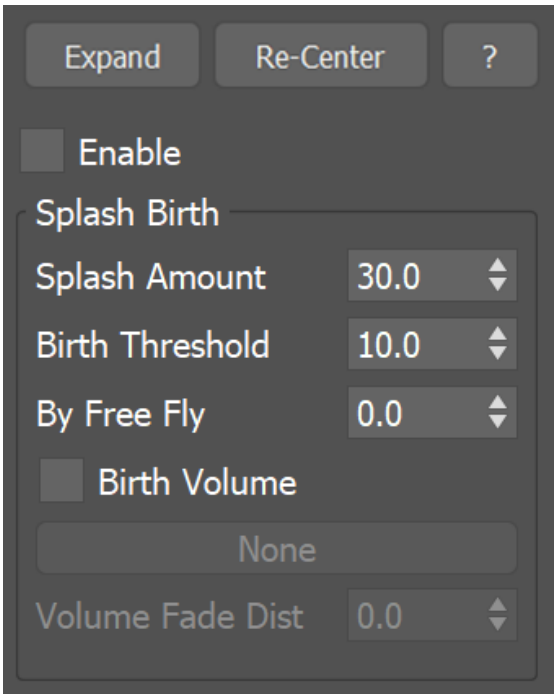
### Parameters

**Expand** – Opens a floating dialog that contains the selected rollout and automatically folds the command panel rollout.

**Re-Center** – Resets the position of the floating rollout.

**?** – Opens up the help documents for the Liquid Splash/Mist.

**Enable** | *splashes* – Enables the birth and simulation of **Splash** and **Mist** particles.




## Splash Birth

**Splash Amount** | *spbirth* – Controls the amount of splash relative to the liquid. This value determines how many splash particles are equal to a single liquid particle in volume. This value also affects the particle size, where larger values decrease the size of splash particles. Note: This parameter is the nearest in functionality to the "birth rate" parameter in previous versions. *For more information, see the Splash Amount example below.*

**Threshold** | *spbthres* – Controls the condition for splash birth. The lower this value is, the more places on the liquid surface will produce splash.

**By Free Fly** | *spfreefall* – How likely a free falling or flying liquid particle will turn into splash. At 0.0, free falling/flying liquid will not convert into splashes. The main usage of values above 0.0 is in waterfall simulations. It is not recommended that this parameter be changed to a value other than 0.0 for ocean simulations because this might create splashes over the ocean surface or even in air pockets under the water surface. *For more information, see the By Free Fly example below.*

**Birth Volume** | ***usespbconfgeom****, spbconfgeom* – When enabled, allows the splash to be born naturally by the simulation only inside a specified geometry object. The splash born inside the Birth Volume can travel outside the volume without a problem. The difference between this approach and spawning splash inside a volume manually from a Source object is that using a Birth Volume, the splash birth will follow the simulation criteria of the simulation and will look and behave more naturally.

By default, the birth volume geometry is not automatically converted to a non-solid and it will behave as a rigid body in your simulation. In this case, you can still use the **Volume Fade Dist** to expand an area around the object where Foam/Splash births are possible. You can convert the geometry to a non-solid from its Chaos Phoenix Properties in order to allow liquid to exist inside it as well.

Also note that birth volumes will interact with the simulation even if they are hidden. If you wish, you can exclude the Birth Volume from the Scene Interaction rollout.

**Volume Fade Dist** | *spbconfgeomfade* – Controls how far the splash particles will spawn around the specified **Birth Volume** geometry object.

## Mist Birth

**Splash to Mist** | *spsplit* – Controls how fast the splash particles are converted into mist. This value determines the portion of the splash volume that is turned into mist after traveling one meter (3.3 feet). During its motion, a splash particle constantly produces mist particles, decreasing the volume of the splash. When the entire volume is exhausted, the splash particle disappears. This parameter controls how fast this happens. If the value is 0.0 the mist is not simulated at all, and if you restore a simulation containing mist, the mist will be deleted.

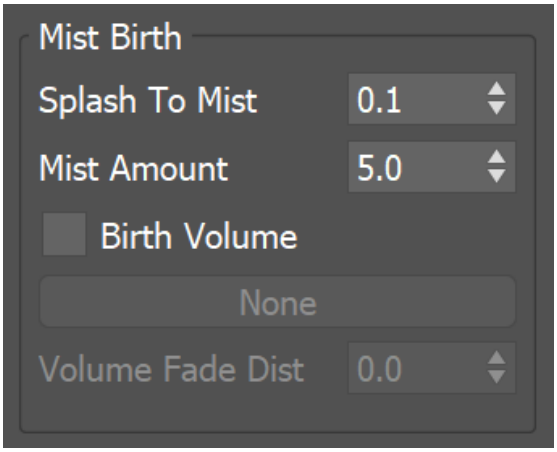
**Mist Amount** | *spmistmult* – Controls the amount of mist. This value determines how many mist particles are spawned by a single splash particle.

**Birth Volume** | *usemistbconfgeom, mistbconfgeom* – When enabled, allows the mist to be born naturally by the simulation only inside a specified geometry object. The mist born inside the Birth Volume can travel outside the volume without a problem. The difference between this approach and spawning mist inside a volume manually from a Source object is that using a Birth Volume, the mist birth will follow the simulation criteria of the simulation and will look and behave more naturally.

By default, the birth volume geometry is not automatically converted to a non-solid and it will behave as a rigid body in your simulation. In this case, you can still use the **Volume Fade Dist** to expand an area around the object where Foam/Splash births are possible. You can convert the geometry to a non-solid from its Chaos Phoenix Properties in order to allow liquid to exist inside it as well.

Also note that birth volumes will interact with the simulation even if they are hidden. If you wish, you can exclude the Birth Volume from the Scene Interaction rollout.

**Volume Fade Dist** | *mistbconfgeomfade* – Controls how far the mist particles will spawn around the specified **Birth Volume** geometry object.

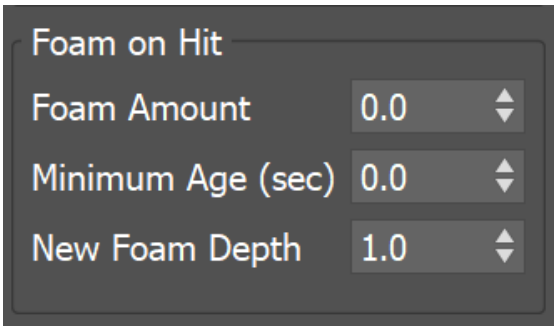



## Foam on Hit

**Foam on Hit Amount** | *sp2foam* – Controls how many foam bubbles are created when a single splash particle enters the liquid volume. This parameter works in terms of probability, so non-integer values can be used. *For more information, see the Foam on Hit example below.*

**Min. Age (sec)** | *spfohminage* – Minimum age (in seconds) for foam production. Only splash particles with a particle age above this limit will produce foam when they hit the liquid surface.

**Depth** | *fohdepth* – When a splash particle hits the liquid surface and produces foam, this parameter specifies how far in front of the splash particle the new foam particles will be born. This parameter is used mostly to create underwater foam that will be colored by the fog color of the liquid.




## Dynamics

**Splash Split** | *splash_split_to_splash* – Controls how quickly the flying splash particles split into smaller ones, creating a trail. If you set it to 0.0, the splash particles would not split at all.

**Splash Air Drag** | *airfr* – The air friction of the splash droplet. A value of 1 represents the air friction of an average rain droplet.

**Mist Air Drag** | *mistdrag* – The air friction of the mist. This value determines not only how fast the speed decreases, but also the influence of the wind. The larger this value, the larger the wind's influence.

**Max Outside Age** | *timeout* – If a particle exits the grid and its age (in seconds) exceeds this parameter, it will be killed instantly. This parameter affects both splash and mist. Note that this is not the time since the particle exited the grid, but the time since the particle was born.

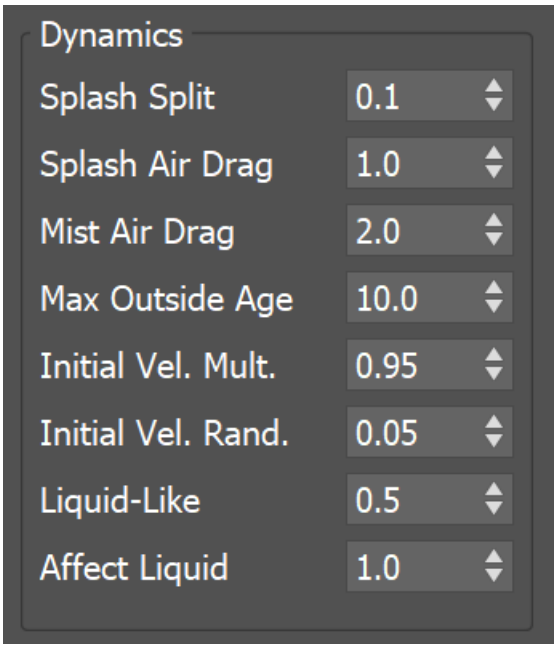
**Initial Vel. Mult.** | *spvmult* – When the splash particles are born, they are initialized with the velocity of the nearest liquid multiplied by this value. This parameter greatly affects the general appearance of the splash. When it is below 1, the liquid droplets fly in front of the splash and produce a clearly visible track.

**Initial Vel. Rand.** | *spvrand* – Controls the randomness of the velocity direction when splash particles are born. This affects the birth of splashes both by liquid particles, and by splitting of splash particles into smaller splash particles. The velocity is randomized equally in all directions, not just in the direction of splash travel, or perpendicular to it.

**Liquid-Like** | *spstick* – Controls the ability of the splash particles to stick to each other, forming different strings and tentacles, so the splash behaves more like liquid and blends better with the behavior of the liquid mass. The higher this value, the larger the acceleration needed to break the connection. Note that larger values will increase the calculation time. *For more information, see the Liquid-Like example belo w.*

**Affect Liquid** | *affectlq* – Controls the conversion of liquid to splash, and splash back into liquid. When this value is not zero, the splash birth causes some of the liquid to disappear, and then when the splash enters the liquid the splash particles are replaced by liquid. A value of 1 with splashes and mist enabled produces a physically accurate simulation where liquid, splash and mist are constantly converted into one another, and the sum of all three at any point during the simulation is the same as the original liquid. This option is good for waterfall simulations but not so much for ocean simulations where you want the ocean surface to be smooth, as it would create bumps in the liquid mesh where splashes fall into the liquid.

When a **Splash** particle is converted into a **Liquid** particle, it inherits the **Default RGB** and **Default Viscosity** values set under the Dynamics rollout.




## Example: Splash Amount

The following video provides an example of **Splash Amount** parameter to show the differences between values of *0.0*, *3.0*, and *10.0*.

Software used: Phoenix FD 4.30.01 Nightly (02 Oct 2020)

Download Example File

## Example: By Free Fly

The following video provides examples to show the differences of **By Free Fly** values of *0*, *0.6*, and *1.0*.

Software used: Phoenix FD 4.30.01 Nightly (02 Oct 2020)

Download Example File

## Example: Foam on Hit

The following video provides examples to show the differences of **Foam on Hit** values of *0.0*, *2*, and *5.0*.

Software used: Phoenix FD 4.30.01 Nightly (02 Oct 2020)

Download Example File

## Example: Liquid-Like

The following video provides examples to show the differences of **Liquid-Like** values of *0*, *2.0*, and *100.0*.

For better visibility in the video, the animation play speed was slowed down after the simulation was completed by reducing the **Input  Play Speed** to **0.25**.

Software used: Phoenix 5.10.00 Official Release

Download Example File

---

Here is an example comparing **Liquid-Like** set to *0* and *3*.

Liquid-Like set to 0

Liquid-Like set to 3

## Note about Splash Particles before Phoenix 3.0

Users of previous Phoenix versions should note an important difference about splash regarding particle size.

In **Phoenix 2.2**, the splash size was controlled directly, like the foam. While foam really needs direct size control, especially for the most popular usage of pouring a glass of beer or soda, the size and volume of a splash should be directly related to the amount of liquid being displaced rather than a user-specified value.

In other words, in order to have a seamless transition between the splash and the liquid, the volume of the splash droplets must be equal to the volume of the liquid converted into splash. **Phoenix 3.0** and newer simulates splashes according to this rule.

Here is an image that may help visualize this process - FLIP Particles Life Cycle.