Foam

This page provides information on the Foam rollout.

Overview

The Foam rollout affects the simulation of **Foam particles**, and enables you to set the conditions for when Foam particles should be born automatically, as well as how the Foam should behave. The simulation of Foam particles can be useful for creating effects like waterfalls, rivers, ocean waves and so forth.

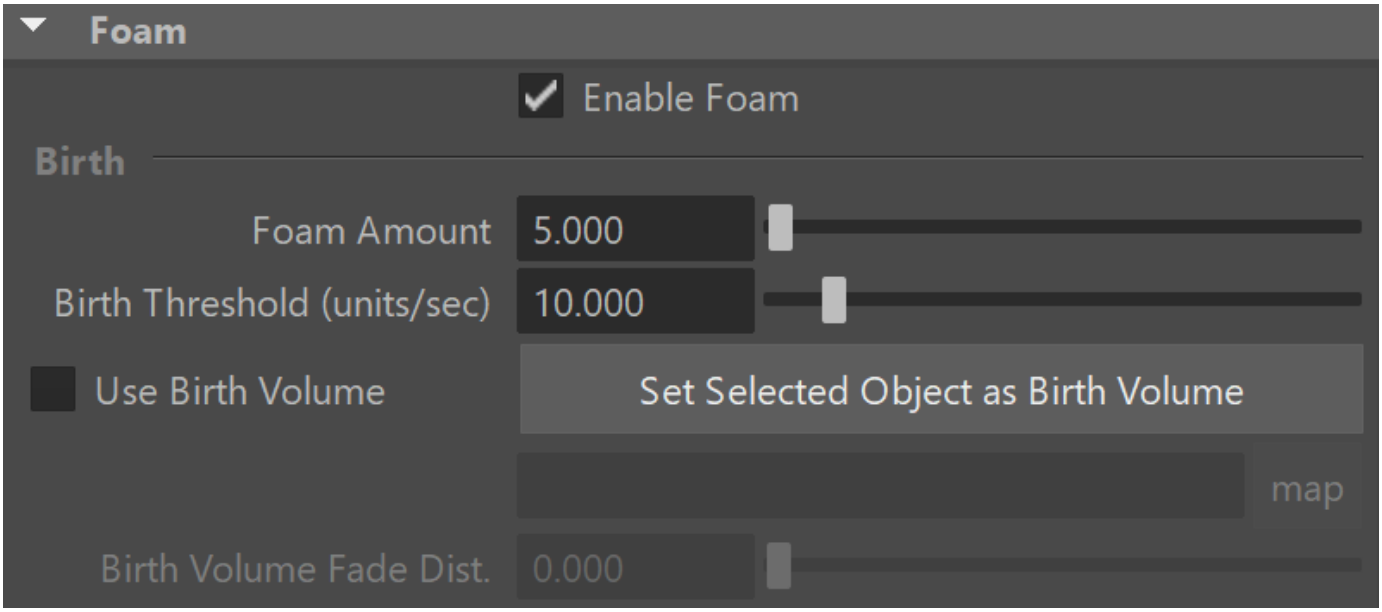
Note that the Phoenix Simulator can simulate different types of particles, including **Liquid particles**, as well as **Secondary Particles** such as **Foam**, **Splash**, **Mist**, and **WetMap** particles. These Secondary Particles exist so that you can achieve a variety of different liquid scenarios.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

Foam particles adhere to several simple rules when simulating: underwater bubbles rise up, bubbles in the air fall down, bubbles can stick to each other, and bubbles can be resistant to external pressure. Regarding the generation of Foam particles, Foam can be born from Liquid particles, Splash droplets, a Source, or a script. Meanwhile, the Foam can disappear when it exits the grid (see the **Max Outside Age (secs)** parameter), or disappear randomly (see the **Half Life** parameter).

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 Foam particles, or how they are created or destroyed, by visiting the FLIP Particles Life Cycle docs page.
Note that unlike shading Liquid particles, in order to **render Foam particles**, you'll need to use the Phoenix Particle Shader to shade them. The Particle Shader enables you to create various fine-tuned appearances for particles such as Foam, Splash and Mist, in order to achieve realistic looking effects.

For example, when rendering Foam particles, you can use the Particle Shader's **Cellular mode** to get a look that is very similar to real foam, which makes it possible to shade foam that is close-up to the camera, and have the results look convincing.
UI Path: ||Select PhoenixFDSim|| **> Attribute Editor > Foam rollout**

Parameters



Enable Foam | *foamEnbl* – Enables the birth and simulation of **Foam** particles.

Birth

Foam Amount | *foamBirthRate* – How many foam particles are born in the places where the **Birth Threshold** condition is met, in particles per cubic centimeter, per second. This parameter does not affect the foam born by splashes.

Birth Threshold | *foamBirthTresh* – Foam is born where the liquid movement is turbulent. If the difference in velocity between two neighbor voxels is higher that this threshold, foam particles should be born. This means that the lower this option is, the more foam will be born and also the more places in the simulator would allow foam to be born. The scale of the option is in units/sec. For more information, see the Birth Threshold example *below.*

Use Birth Volume | *useFoamBirthVolume* – When enabled, allows the foam to be born naturally by the simulation only inside a specified geometry object. This includes foam born according to the conditions set by the **Foam Amount** and **Birth Threshold** parameters of the Foam, as well as the **Foam on Hit** p arameters of the Splash. The foam born inside the **Birth Volume** can travel outside the volume without a problem. The difference between this approach and spawning foam inside a volume manually from a Source object is that using a **Birth Volume**, the foam birth will follow the simulation criteria of the simulation and will look and behave more naturally.

Set Selected Object as Birth Volume - When a polygon mesh and a Phoenix Simulator are selected, the selected object will be used as the **Birth Volume** for the generation of Foam particles in this simulator.

Birth Volume | *foamBirthVolume* – Specifies a geometry object to confine foam particles. To do this, select the simulator and the confine geometry, and hit the **Set Selected Object as Birth Volume** button.

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 **Birth 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 Phoenix FD Extra Attributes in order to allow liquid to exist inside it as well.

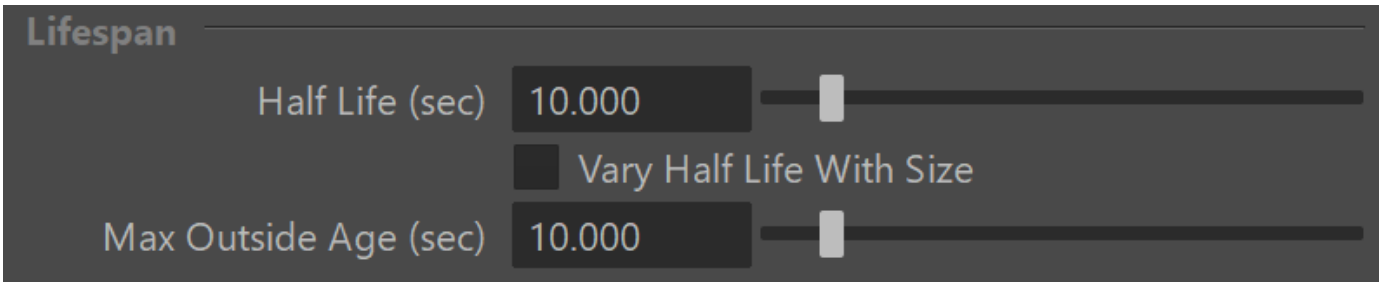
Also note that birth volumes will interact with the simulation even if they are hidden. If you wish, you can exclude the Birth Volume from the Scene Interaction rollout.

Birth Volume Fade Dist | *foamBirthVolumeFade* – Controls how far the foam particles will spawn around the specified **Birth Volume** geometry object, in world units.

Example: Birth Threshold

The following video provides examples to show the differences of **Birth Threshold** values of *50*, *100*, and *200*.

Lifespan

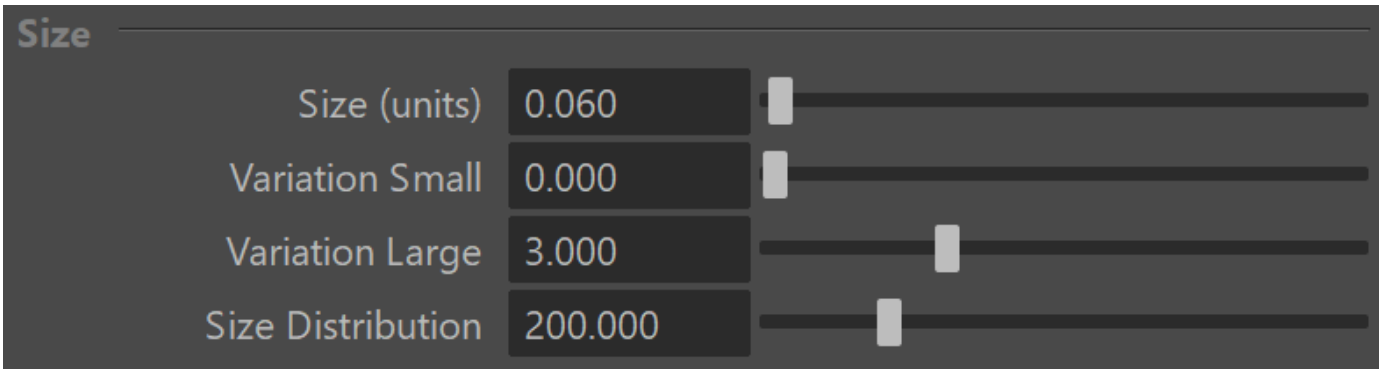


Half Life (secs) | *foamHalflife* – The time required for the particles to reduce to half of their initial count, in seconds. Affects only the foam bubbles above the liquid surface; those inside the volume will not burst until they reach the surface. You can force this process by increasing the **Rising Speed**.

Vary Half Life With Size | *foamHalfLifeVary* – In nature, not all the bubbles have the same half life. Larger bubbles burst quicker than smaller ones, and bubbles inside foam live longer. When this option is enabled, the **Half life** of each bubble is modified taking into account its size and its surroundings.

Max Outside Age (secs) | *foamOutsideLife* – If a particle is outside the grid, and its age (in seconds) exceeds this parameter, the particle will be killed instantly. Note that this is not the time since the particle exited the grid, but the time since the particle was born.

Size



Size (units) | *foamSize* – Specifies the size of the bubbles.

Larger bubbles rise faster while they are underwater, while smaller ones rise more slowly.

Variation Small | ***foamDownVary*** – Specifies how much smaller the bubbles can be, relative to the **Size** parameter. *0* means that all bubbles are always at least as big as **Size**. *1* means that the smallest bubbles are two times smaller than **Size**. *10* means that the smallest bubbles are 11 times smaller than **Size**. *For more information, see the Variation Small example below.*

Variation Large | *foamSizeVar* – Specifies how much larger the bubbles can be, relative to the **Size** parameter. *0* means that all bubbles are never larger than **Size**. *1* means that the largest bubbles are two times larger than **Size**. *10* means that the largest bubbles are 11 times larger than **Size**. You can also use the **Size Distribution** parameter to control the balance between the number of small and large bubbles. *For more information, see the Variation Large example below.*

Size Distribution | *foamSizeDistr* – Specifies how many times the amount of bubbles with radius **Size** exceeds the amount of the largest bubbles. This option has no effect if **Variation Large** is 0. Setting **Distribution** to *0* means that all sizes will be equally distributed. *100* means that there will be 100 times more bubbles with radius of **Size** than the largest ones. *For more information, see the Size Distribution example below.*

Example: Variation Small

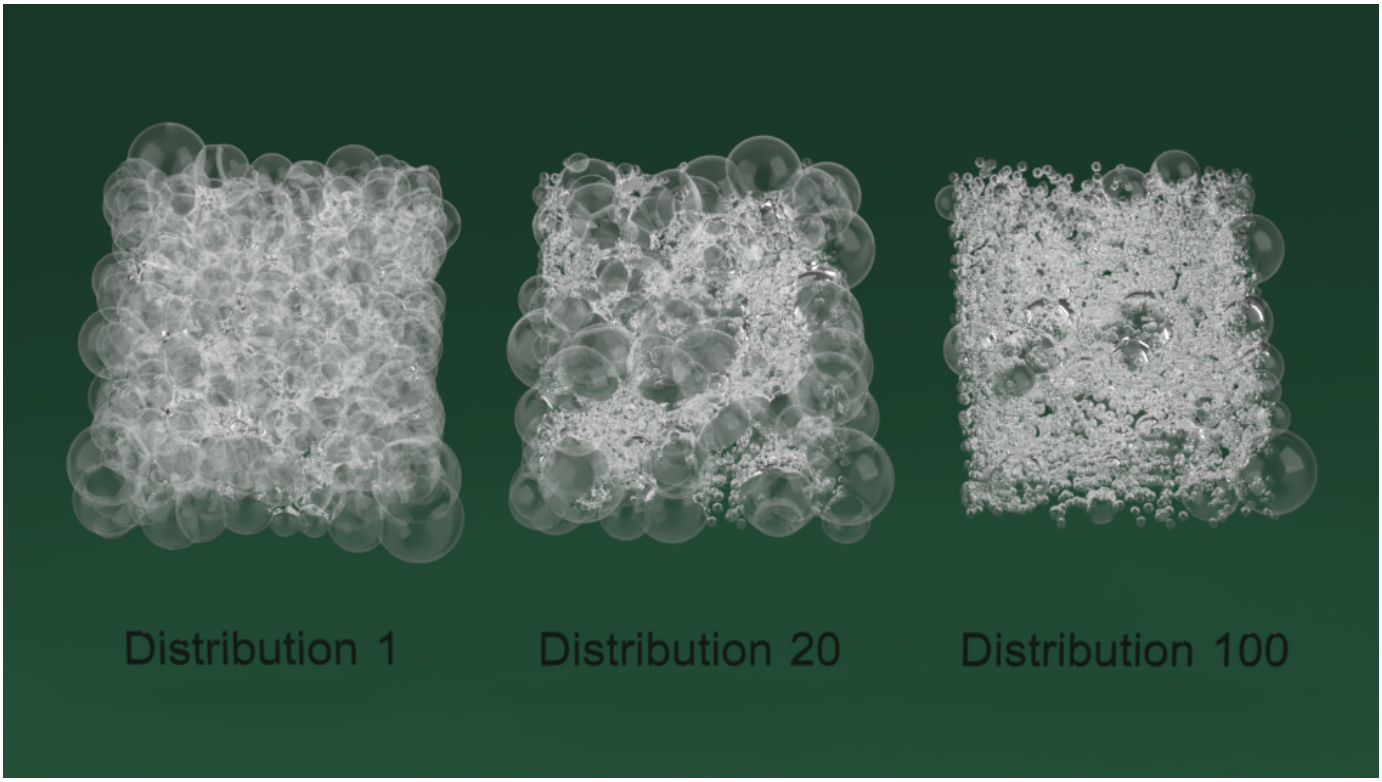
The following video provides examples to show the differences when **Variation Small** is set to *0*, *1*, and *50*.

Example: Variation Large

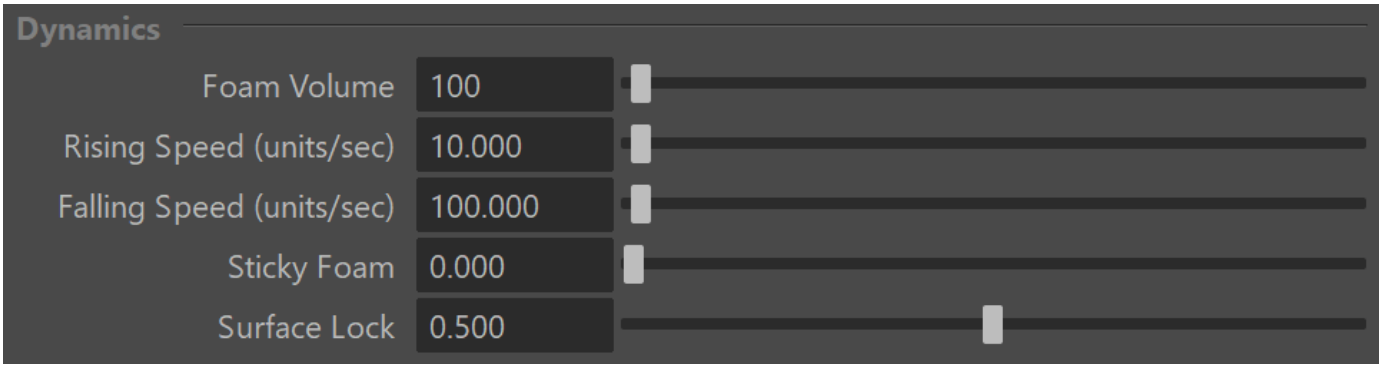
The following video provides examples to show the differences when **Variation Large** is set to *0*, *3*, and *6*.

Example: Size Distribution

The image below provides more details on the differences between **Size Distribution** values of *1*, *20*, and *100*.



Dynamics



Foam Volume | *foamVolCycles* – Controls the internal interaction between bubbles (bubble-to-bubble interaction). This option is used when the foam should have a volume. It forces a proper distance between the bubbles and keeps them stuck together. This parameter controls the number of interactions per second. Higher values result in better preservation of the foam's volume. This parameter has linear growth order. In other words, the time taken for calculation is longer when the value is higher. *For more information, see the Foam Volume example below.*

Rising Speed (units/sec) | *foamRiseLimit* – Affects only the foam particles which are inside the liquid volume. This is the maximal speed of ascent of the average-sized bubbles. Bubbles larger than **Size** rise faster and smaller bubbles rise more slowly. Note that bubbles inside the liquid volume are carried by the velocity of the liquid, and also affected by this parameter. The scale of the option is in units/sec. If the **Rising Speed** is too high, foam particles will shoot past the liquid surface into the air, like bubbles from a carbonated drink - you can compare the value of this option to the height of the Simulator to get an idea of it's too high or too low. *For more information, see the Rising Speed example below.*

Falling Speed (units/sec) | *foamFallLimit* – Affects only the foam particles which are outside of the liquid volume. This parameter controls the drag force of foam flying through the air. The Falling Speed is also affected by the **size** of the individual bubbles. Lowering this parameter towards zero would cause foam to rapidly slow down when flying in the air, or hang in the air for a long time until it falls down. Note that this parameter affects only foam which is outside the liquid volume, while foam inside of the liquid mesh or which is mixed up with flying Liquid particles would have the same velocity as the liquid. In the second case, if the **Falling Speed** is too low, the foam might abruptly slow down at the moment it separates from the liquid that carries it. The scale of the option is in units/sec. *For more information, see the Falling Speed example below.*

The **Falling Speed** parameter affects how far a flying foam particle may travel. A higher **Falling Speed** may lead to separate bubbles thrown far away from the simulator which leads to huge bounding box and thus slower rendering.

Therefore, this parameter can dramatically affect the rendering speed.

Sticky Foam | *foamSticky* – Controls the ability of the bubbles to stick to a geometry. This option requires that the **Foam Volume** is greater than zero.

Surface Lock | *foamSurfLock* – Affects the bubbles placed on the liquid surface. It controls the influence of the liquid movement over the surface bubbles. This parameter can be used together with the Foam Pattern options.

Example: Foam Volume

The following video provides examples to show the differences when **Foam Volume** is set to *0*, *500*, and *1000*.

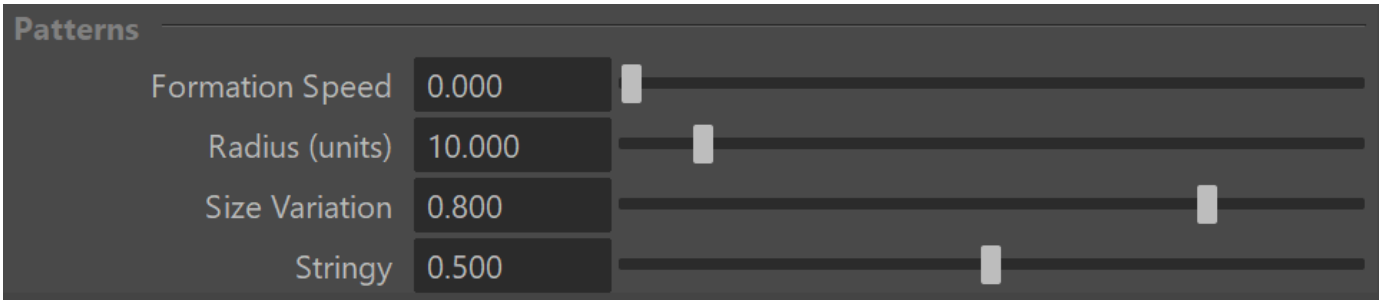
Example: Rising Speed

The following video provides examples to show the differences when **Rising Speed** is set to *0*, *20*, and *100*.

Example: Falling Speed

The following video provides examples to show the differences when **Falling Speed** is set to *10*, *35*, and *100*.

Patterns



Formation Speed | *fptrn* – Controls the rate of formation of foam patterns. In nature, these are caused by liquid flows rising to the surface and pushing the foam aside. *For more information, see the Formation Speed example below.*

Radius (units) | *foamPatternSize* – The average radius in scene units of a single circular pattern core.

Size Variation | *fptrnszvar* – Creates several larger patterns and many smaller ones. If set to 0, all patterns will have equal sizes.

Stringy | *fptrnstringy* – Higher values will cause the formation of thin, long strings between the patterns.

Example: Formation Speed

The following video provides examples to show the differences of **Formation Speed** values of *0*, *0.5*, and *1.0*.