Interactions Between Simulators

Overlapping Simulators

In Phoenix, a Simulator sees other Simulators in the scene just like any other deforming mesh geometry. The loaded cache data into a Simulator represents the Isosurface of the Simulator that the other Simulators see. Simulators are Solid by default, and this way they are obstacles to other Simulators which intersect them, but you can also make a Simulator non-Solid. You can easily preview the interaction surface of each Simulator in the viewport if you enable its Show Mesh option from the Preview rollout - this will be the shape that other simulators will interact with. Note that this applies even if the Simulator would render in any other mode, not just Isosurface or Mesh.

A Simulator can also be an emitter in another simulation. If you pick it into a Source, you can make it emit fluid in any Emit Mode. This way you can have a liquid simulation in one Simulator, and then set it on fire in a second Simulator.

A Simulator can also be used by a Body Force to attract or repel the fluid of another simulator, and also can be excluded or included from other Simulators' Interaction list. Note that by default all Simulators are included in one another's simulations, just like any other geometry in the scene, so if you don't want a Simulator to affect another one, you have to exclude it explicitly.

Simulators interacting as obstacles

When you simulate overlapping Simulators, you would often need to run the Simulators in a certain order. For example, you could have two Liquid Simulators, where "Sim1" would be a heavy liquid that would fill the bottom of a container first, and then "Sim2" would be a lighter liquid that will not blend with the liquid from Sim1, but instead will spill over it and float on top of it. In this case you need to run Sim1 first, and then when you run Sim2 it would by default see the liquid of Sim1 as an obstacle. This way the new liquid would spill over Sim1's Isosurface. Once you already have a cached simulation for Sim2, you must consider that Sim1 now would also see the liquid from Sim2 as a Solid obstacle by default, so if you run Sim1 for a second time, without changing any settings, it would produce a different simulation from the first time it was run. In order to avoid this, you have to make Sim1 ignore Sim2's Isosurface - so you could exclude Sim2 from the Interaction rollout of Sim1. So then, when you want to change how Sim1 behaves, you would need to first run Sim1 and then also run Sim2 after that, so Sim2 would interact with the new Isosurface of Sim1.

When you exclude simulators from interaction, you can use either Exclude or Include mode under the Interaction rollout, in case you get circular reference errors. If you need Include mode, you have to include all sources, all of their emitters too, all obstacles, and all forces affecting a simulator. Also note that if you have two interacting simulators, each with its own source, you might also need to exclude the source for Sim1 from Sim2 and vice versa. Otherwise, if the emitters from a source intersect both simulators, they would always emit fluid when you simulate any of the two simulators, and this might not be a desired effect.

Using one Simulator as an Emitter in another Simulator

The same way that one Simulator can be used as a plain obstacle to another, it can also be used to emit fluid from its Isosurface into a different Simulator. You just need to add a Source and select the first Simulator in it. This way, your first Simulator can run a Liquid simulation, and then be used as an emitter in a Fire simulation, so the liquid mesh would ignite in the second Fire/Smoke Simulator.

Transferring fluid between Simulators using a Cascade Connection

A more advanced example of interaction between two simulators would be a cascade connection, where the fluid flows from one simulator into another. You can use cascade setups for simulations with irregular shapes where using just one Simulator grid would leave a lot of empty space and consume an unneeded amount of RAM and processing power. Additionally, if the simulation can be broken into pieces that should run one after another, then the cascade simulation would also be a good choice - this way you can iterate on the first Simulator until it's right, then iterate on the second Simulator for the continuation of the effect, and so on.

Cascade simulations are handled differently for Fire/Smoke Simulators and Liquid Simulators:

- A Liquid Simulator can be connected in a cascade setup to the previous Liquid Simulator in the sequence using the Cascade Simulator connector in the Grid rollout. The liquid flows from the Cascade Simulator to the Simulator that points to it. The Simulators must be run in this order as well. The only requirement for this setup is that the connecting simulators overlap. They can be oriented in any way and can have different resolutions. Many Simulators can be connected in a cascade chain, and many simulators can get fluid flown in from the same Cascade Simulator. Note that when rendering, each simulator will produce its own Mesh or Isosurface, so when using a refractive material, there could be visible seams between the different Simulators.
- Fire/Smoke Simulators require more manual work because they don't simply have fluid Isosurfaces, but their volumes are irregular and have varying density. A Source can be used to make the first Fire/Smoke Simulator an emitter for the second one. The Source would emit in Volume Brush mode the kind of grid channels simulated by the first Simulator from the shape of the first Simulator's Isosurface. However, simply emitting from the Source would create constant density inside the volume, and would ignore the internal detail and density of the emitter Simulator. This can be solved by modulating the emission of each Source channel by a separate Grid Texture, each Grid Texture reading the corresponding channel from the emitter Simulator. This would imprint the exact fluid channels of the emitter Simulator into the destination Simulator.