

# Liquid Rendering

## Overview

The rendering process in Chaos Phoenix is separated from the simulation process, because simulated caches contain only simulation data, and no render settings.

However, for ease of use, the Liquid Simulator object also contains a Rendering rollout, which provides flexible options for rendering the simulator grid's content.

The Rendering rollout offers multiple render modes, that can be divided into two types: **Surfaces** and **Volumetric**.

Liquid simulations are typically rendered using one of the surface render modes. Meanwhile, volumetric modes are typically used for rendering fire and smoke.

**Surface render modes** generate a mesh surface, which is based on the channel specified in the **Surface Channel** parameter.

This is especially useful for liquid simulations, where by default, the **Grid Liquid Channel** is exported during simulation, which enables the simulator to voxelize **Liquid Particles** into a **Grid Channel**. This way, Liquid Particles can be used to indirectly generate a liquid surface, that can be rendered as a mesh using one of the surface render modes, such as **Mesh**, **Isosurface**, **Cap Mesh** or **Ocean Mesh**.

The meshed liquid then behaves just like any regular geometry, meaning you can assign 3ds Max or V-Ray materials to the Simulator, and there is no need for a dedicated shader. The Rendering rollout also contains additional controls for this meshing process, to customize the appearance of the surface.

Since Phoenix is very flexible, it also enables you to render liquid simulations using a volumetric render mode, to create more advanced types of effects. When rendering as a mesh, Liquid Particles are shaded by a material assigned to the Liquid Simulator object.

However, **secondary effect particles** such as **Foam**, **Splash**, **Mist** and **WetMap** particles must be shaded using the Phoenix Particle Shader.

The Particle Shader can shade a specified Particle System as either Points, Bubbles, Cellular, Splashes, or Fog, depending on the mode you select.

For even more advanced control over the shading, you can use the **Phoenix Grid Texture**. It reads from the simulation's Grid Channels to generate a procedural texture, which can then be used to shade the simulation wherever colors are needed.

It can also be plugged into the texture slots of a material. For example, if you want to mix together liquids with multiple **RGB colors** emitted from different **Liquid Sources**, the Grid Texture can be used to read and transfer the RGB colors to the Liquid mesh's material for shading.

Similarly, a **Phoenix Particle Texture** can be used to read particles and color their positions. When used to read **WetMap** particles, it can act as a mask to blend between two materials, for example, a wet material and a dry surface material. This way, geometry covered by WetMap particles can appear wet, and the rest of the geometry can appear dry.

UI Path: ||Select Liquid Simulator object|| > **Modify panel > Rendering rollout**

## Actions

Render settings are not stored within the caches themselves, so if you want to use the same render settings for another simulator or project, there is the option to save and load them as **Phoenix Render Presets** in the ".tpr" file format.

**Render Presets...** – Opens a menu for loading and saving different presets. The following options are available:

- Load from File...
- Save to File...
- Default .aur Render Settings
- Liquid .vdb from Houdini

## 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 Rendering.

**Mode** | *rendMode* – Specifies the method for visualizing the grid content.

**Volumetric** – Visualizes the content as a standard 3ds Max atmospheric. This method is used mostly for fire and smoke.

**Volumetric Geometry** – This method requires V-Ray. It produces the same result as the **Volumetric** option by using procedural geometry made up from multiple transparent layers. Used when rendering fire/smoke for exporting deep images and render elements such as normals, velocity, multi matte, etc. which would not be available in **Volumetric** mode. For more information on which render elements are supported in Volumetric and **Volumetric Geometry** mode, see [V-Ray Render Elements Support](#).

**Approximate** and **Approximate+Shadows** options for the **Scattering** parameter in the Smoke color window are not supported in **Volumetric Geometry** mode.

For a complete list of the supported **Render Elements** in both Volumetric and Volumetric Geometry mode, please check the [V-Ray Render Elements Support](#) page.

**Volumetric Heat Haze** – This method requires V-Ray. It produces the same result as the **Volumetric Geometry** option, and also adds a heat haze effect when used with the **Heat Haze** parameter. Note that you might need to increase the **Max depth** of a V-RayMtl with refraction in case it intersects with the Heat Haze shader.

**Isosurface** – This method requires V-Ray. It produces a procedural **isosurface** without polygons at render time using the **Isosurface Level** option. Compared to the **Mesh** mode, the result is always smooth but will take longer to render. In case in **Mesh** mode your mesh is too jagged and edgy, and smoothing it out is too slow or impossible, this means you should switch to **Isosurface** mode instead.

**Mesh** – The content is converted into a standard 3ds Max mesh using the options in the Mesh Smoothing section. This mode is mostly used for liquids but can also be applied to thick smoke using a scatter material or to plumes of smoke to create effects such as large underwater bubbles.

**Ocean Mesh** – The grid content is extended to a flat area, fitting the camera's view. In most cases, this mode is used with a displacement texture such as the [Phoenix FD Ocean Texture](#).

**Cap Mesh** – Only the upper liquid surface is rendered. This mode can be used for swimming pools and other placid liquid surfaces.

The ocean surface can be generated only when the liquid touches the sides and the bottom of the grid, which act as a container for the liquid. The detail of the mesh extension around the simulator depends on the camera resolution - for each pixel of the viewport or the rendered image, one or several polygons are generated, depending on the **Ocean Subdivisions** option.

Also note that if you use a material with fog color for the ocean mesh, and you have particles submerged below the ocean surface which you render using a Particle Shader, you need to also place a geometry that would serve as a bottom, or you could get flickering and darker rendering of the particles. For more info, see the **Render as Geometry** option of the Particle Shader.

**Cutter Geom** | *usegizmo, gizmo* – When enabled, rendering will occur only inside the selected geometric object's volume. Note that the **Cutter Geom** will not work when the render **Mode** is set to **Volumetric Geometry**.

**Invert Cutter** | *invgizmo* – When enabled, rendering will occur only outside of the render cutter. This is not the same as a cutter with inverted geometry because any rays that do not intersect the cutter will be shaded as well.

If using a **Render Cutter** for a liquid pouring into a glass or otherwise contained into another refractive object, you may need to set the **Mode** to **Isosurface**. By default, the mode is set to **Mesh** which may produce artifacts in the rendered image.

**Surface Channel** | *sarg* – Specifies the channel that will define the surface of the fluid. It is used for solid rendering and displacement.

**Texture** - the values of a custom texture will define the mesh surface. You can see how this works in [this How-to video](#).

**Liquid** - the Liquid channel will define the liquid surface. Liquid is typically in the range 0-1 for Liquid simulations.

**Smoke** - the Smoke channel will define the liquid surface. Smoke is typically in the range of 0-1 for Fire/Smoke simulations.

**Speed** - the Speed channel will define the liquid surface. Speed channel output has to be enabled for this to work. Speed is calculated as the length of the velocity vector for each voxel.

**Fuel** - the Fuel channel will define the liquid surface. Fuel channel output has to be enabled for this to work.

**Viscosity** - the Viscosity channel will define the liquid surface. Viscosity channel output has to be enabled for this to work.

**Texture** | *stex* – If the **Surface** channel is set to **Texture**, this slot specifies the texture. In **Mesh**, **Ocean Mesh**, **Cap Mesh** and **Isosurface** render modes, the selected map will completely replace the cache files that have been loaded, if any. For more information on texture mapping in Phoenix, please check the [Texture mapping, moving textures with fire/smoke/liquid, and TexUVW](#) page.

**Invert Liquid Volume** | *solidbelow* – By default, grid values above 0.5 are assumed to be liquid, and values below 0.5 are assumed empty. When enabled, the convention switches to the opposite such that values above 0.5 are assumed empty, and values below 0.5 are assumed to be liquid.

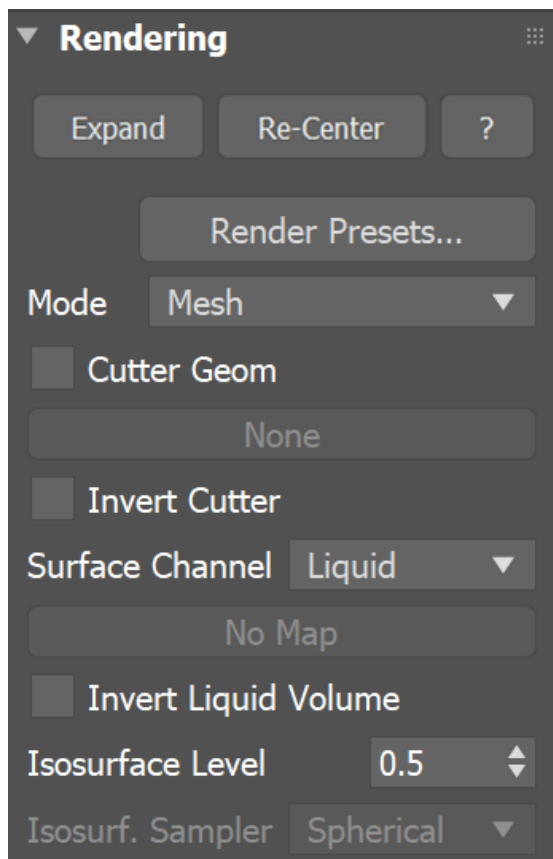
**Isosurface Level** | *surflevel* - Allows you to specify a threshold value for the generation of the liquid surface. Grid cells below this value will be ignored. By default, the Isosurface Level is set to 0.5 and should only be modified if there is flickering in the generated geometry. Isosurface Level is used only in **Isosurface**, **Mesh**, **Ocean Mesh** and **Cap Mesh** Modes.

**Isosurf. Sampler** | *sampler* – Determines the blending method between adjacent grid voxels. Used to balance between rendering speed and quality. This parameter is only used when **Mode** is set to **Isosurface**.

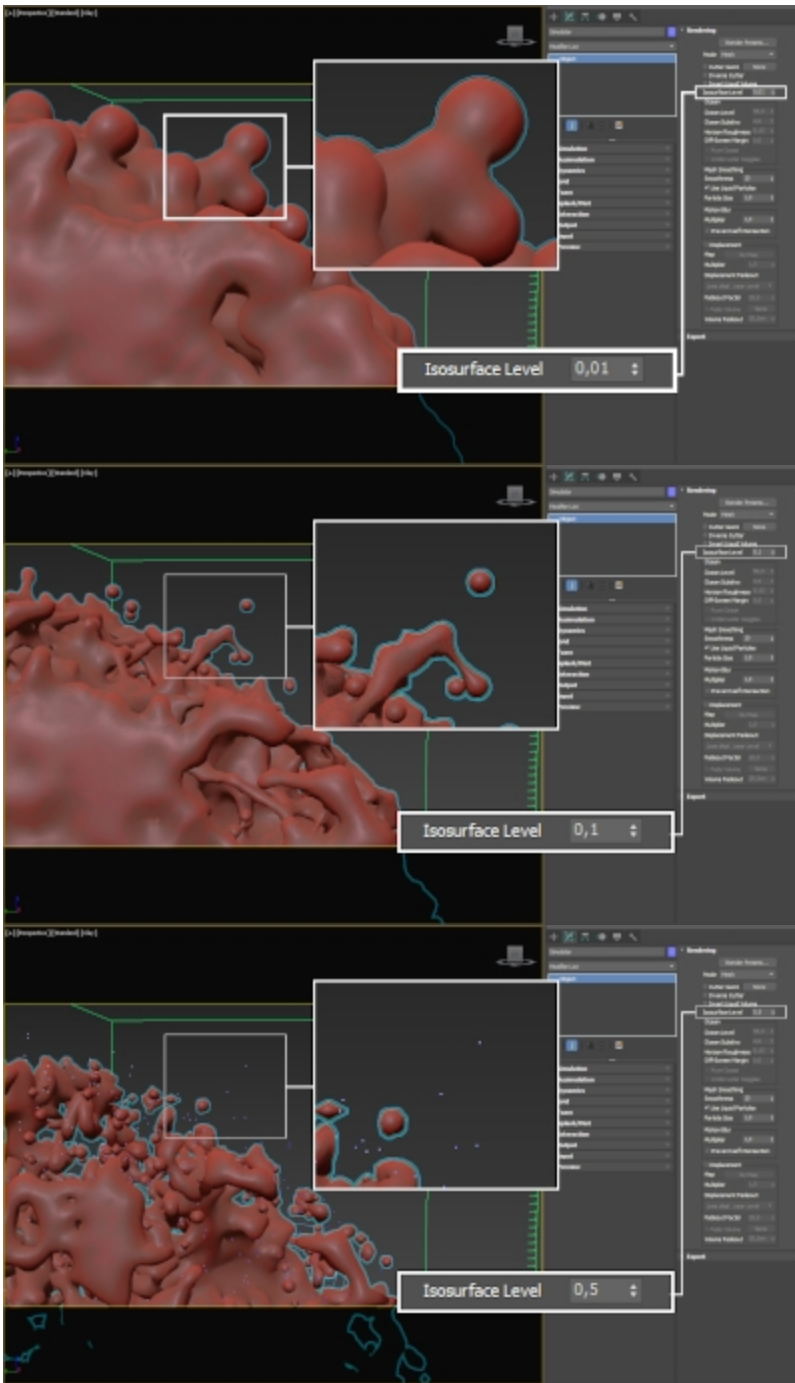
**Box** – Displays voxels as cubes. There is no blending between neighbor voxels. This is the fastest mode.

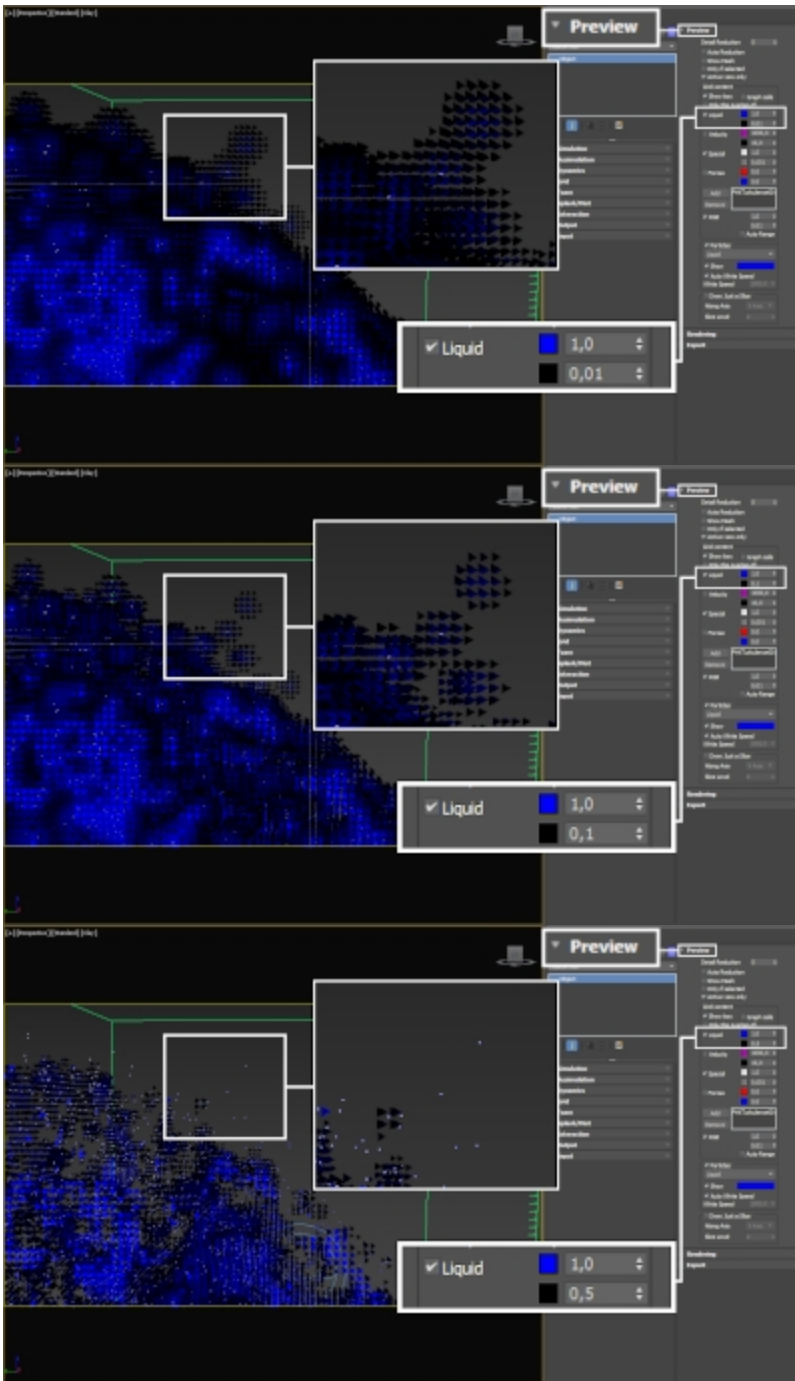
**Linear** – Linear blending occurs between neighbor voxels to smooth out the fluid's look. Sometimes this mode may unveil the grid-like structure of the fluid. Up to 20-30% faster than the **Spherical** option.

**Spherical** – Uses special weight-based sampling for the smoothest looking fluid. With increasing resolution, the visual advantage of this method over the **Linear** method becomes less noticeable.



The proper value for the **Isosurface Level** parameter depends on the numerical range of the surface channel. For example, Phoenix liquids are kept in the range of 0 to 1. A value of 0 means there is no liquid in a certain voxel, and a value of 1 means the cell is 100% full of liquid. Values in between indicate a certain mixture of air and liquid. For such cache files, an **Isosurface Level** value of 0.5 is best for visualizing the surface between the air and liquid. Imported caches from Houdini, on the other hand, use positive and negative values to indicate whether a voxel is inside or outside the liquid volume, so a correct "halfway" **Isosurface Level** value would be 0.0. Please check the Grid Channel Ranges page for information about other grid channels.





## Ocean

**Ocean Level %** | *oceanlevel* – Used with the Ocean Mesh and Cap Mesh rendering modes. Specifies the water level as a percentage of the total grid height.

- **Ocean Level** is a rendering setting which is closely related to the **Initial Fill Up** simulation setting in the Dynamics rollout.
- Both should initially be set to equal values, and the **Ocean level** should be adjusted after the simulation is done, if needed. This has to be done manually because the liquid surface might bounce up or down in the first few seconds until it settles.
- Depending on a variety of factors, the resulting surface might be a little below or a little above the **Initial fill up** value, and this cannot be automatically predicted.

- The **Ocean level** should remain constant and should not be animated for the duration of the render sequence, otherwise the horizon would start moving up and down.

**Border Fade %** | *oceantransitionzone* - The width of a zone inside the Simulator Grid where the mesh surface level would smoothly transition into the displaced waves outside the Simulator. The value is a % of the Simulator width. Increase the value if you need a smoother transition. This option can be especially useful for merging tall waves created by a Wave Force inside the Simulator, with tall displaced waves outside the Simulator.

**Ocean Subdivs** | *meshsubdiv* – Used with the **Ocean Mesh** and **Cap Mesh** rendering modes. When generating the far areas of the surface, this determines how many vertices will be generated for each pixel of the image. Like with V-Ray subdivisions, the square of the parameter value is used. For example, if you increase the subdivisions twice, the vertices count will grow four times. *For more information, see the [Ocean Subdivs](#) example below.*

Increasing the **Ocean Subdivs** may dramatically increase the amount of consumed RAM.

**Horizon Roughness** | *oceanhorizrough* – Controls the roughness of the distant ocean surface near the horizon. Adds more detail to the waves, which is especially helpful when using a highly reflective material for the ocean surface. Increasing this value can potentially produce visible noise when rendering an animation. To counter this effect, increase the **Ocean Subdivs** parameter.

**Off-Screen Margin (%)** | *oceanextramargin* – The ocean is generated only in the camera view, which can lead to problems when using camera motion blur, using reflections, using refractions with **Underwater Goggles** enabled, or when the ocean casts shadows on objects underwater. This option allows you to extend the ocean further from the borders of the camera view in order to solve such issues. The value is a percentage of the image size.

**Pure Ocean** | *pureocean* – Creates a flat ocean surface up to the **Ocean Level** height. It does not need loaded caches and if there are any, it ignores their content, so no simulation details will show. Thus changing frames and generating the ocean surface is very quick. This allows you to preview the behavior of the [Phoenix FD Ocean Texture](#) when Displacement is enabled or if you want to set up your texture for the Wave force. The option is available for both preview and rendering in **Ocean Mesh** or **Cap Mesh** modes. During preview, it requires the **Show Mesh** option to be enabled in the [Preview](#) rollout.

**Underwater Goggles** | *uwglasses* – This option is designed to be used when the camera is placed under the water in **Ocean Mesh** or **Cap Mesh** mode. When enabled, a surface gets automatically added in front of the camera, using the same material that is assigned to the Simulator, and this way mimics the effect of real life underwater goggles. This way the ocean volume receives the Fog color from the material assigned to the Simulator, and the field of view shrinks when the index of refraction of the material is above 1. *For more information, see the [Underwater Goggles](#) example below.*

**oceanpriority** – Script parameter with a default value of 0. This parameter can be used for controlling which is the primary ocean container when several ones are merged together. The Simulator with the highest priority will provide the material and the rendering settings (the Ocean Level, the Ocean Subdivs, etc.) for the merged ocean. The actual combined mesh is not affected by the ocean priority.

Example:

Select a Simulator that you want to have more priority and in the MaxScript Listener type: **\$oceanpriority = 3**

The Simulator with the highest priority in the scene will be the one whose material and ocean settings will be used by the combined ocean mesh.

**oceanforcerenddetail** – Script parameter that allows to build the viewport Ocean with the rendering detail, otherwise a simplified version will be used for the viewport preview.

Note that higher detail will impact the viewport performance.



Ocean

Ocean Level %	50.0	▲▼
Border Fade %	10.0	▲▼
Ocean Subdivs	2.0	▲▼
Horizon Roughness	1.0	▲▼
Off-Screen Margin	0.0	▲▼

☐ Pure Ocean

☐ Underwater Goggles

### Example: Ocean Subdivs

---

2 subdivs  
8 subdivs



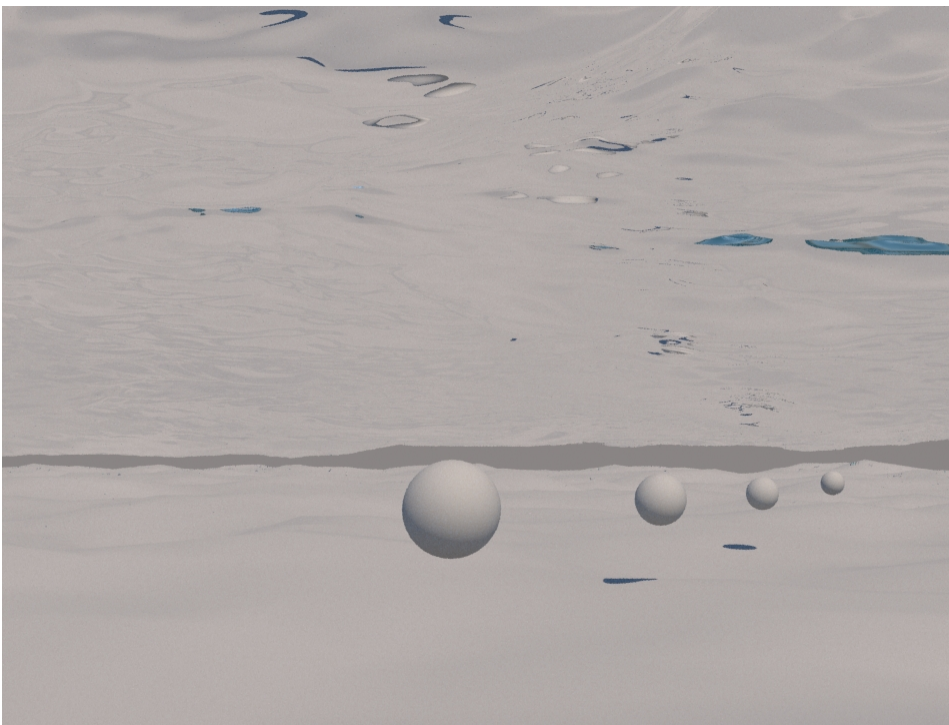


### Example: Underwater Goggles

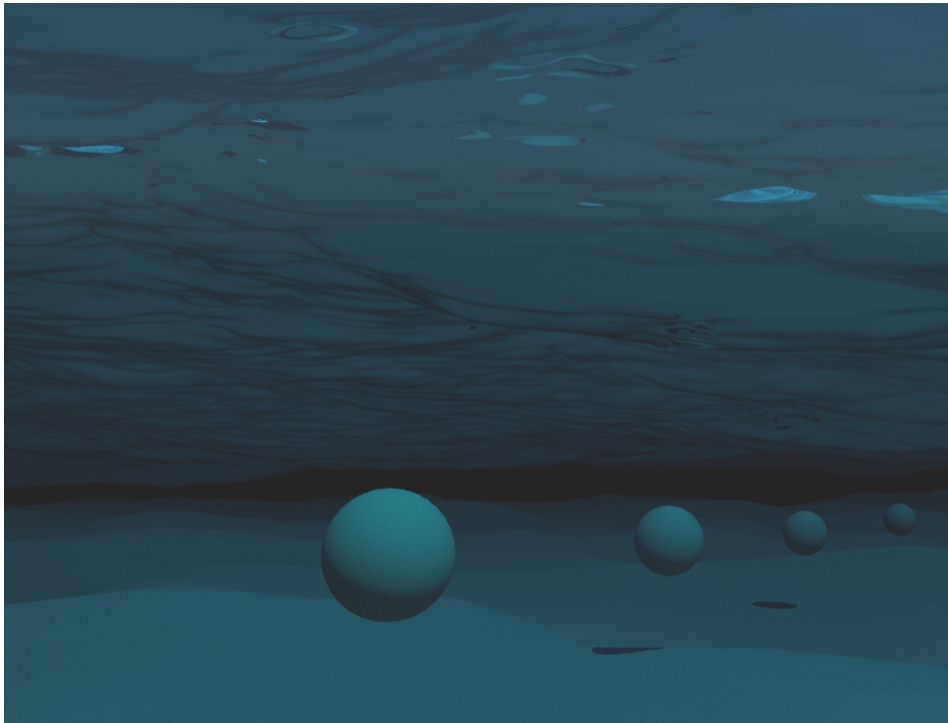
---

Notice how the field of view shrinks when the option is enabled, due to using **Refraction IOR = 1.33** in the Simulator's V-RayMtl.

Off  
On







## Mesh Smoothing

---

**Smoothness** | *smoothmesh* – Specifies the number of smoothing passes. The higher the value, the smoother the result, but the mesh will require more time to calculate. Used when **Mode** is set to **Mesh**, **Ocean Mesh**, or **Cap Mesh** to reduce the roughness of the mesh.

**Smooth Normals** | *smooth\_normals* – Smooths the normals of the mesh in order to get an even smoother looking result, even if the mesh is with low resolution. However when the Ocean displacement is used to add fine details the smoothing of normals should not diminish that and can be turned off.

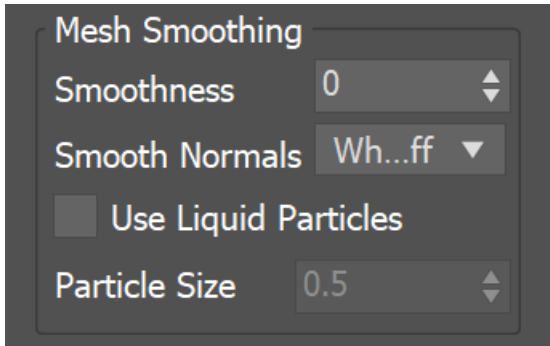
**Always** – The mesh normals will be smoothed always.

**When displacement is off** – The mesh normals will be smoothed only when the Ocean displacement is off.

**Never** – The mesh normals will not be smoothed.

**Use Liquid Particles** | *useprt* – Enables particle-based smoothing of the mesh. It requires Liquid particles to be simulated and exported to the cache files. This method overcomes the limitations of the basic smoothing without particles, which can flicker in animation and cause small formations in the mesh to shrink.

**Particle Size** | *prtsz* – Used to make the liquid thicker or thinner. Works only when **Use Liquid Particles** is enabled. This parameter specifies the distance from the mesh surface to the particle centers, in voxels.



## Motion Blur

To render your simulation with Motion Blur, you need to enable Velocity channel export from the Output rollout of your simulator.

When rendering liquids, the Motion blur of the mesh is obtained by shifting each vertex along the velocity by the shutter time. If rendering a Liquid simulation with secondary particle effects such as Foam, Splashes or Mist, you would also have to enable Velocity export for each particle system under the Output rollout Output Particles section.

**Multiplier** | *mbmult* – Specifies a multiplier that affects the strength of the motion blur. This value can be a negative number.

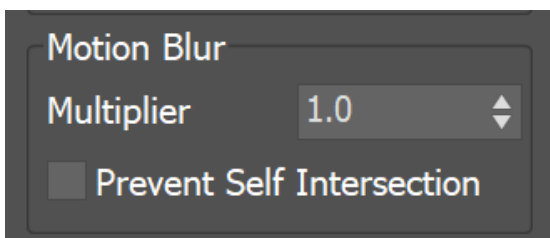
**Prevent Self Intersection** | *mbself* – In all **Mesh** modes, some motion blurred vertices might penetrate the opposite side of the geometry. When enabled, this option prevents such situations. The self-intersection analysis is expensive, so enable this parameter only when an intersection is obvious.

Phoenix meshes are motion blurred in a different way than regular transforming and deforming geometries. When rendering regular meshes with motion blur, the entire mesh is moved along its transformation path back and forward in time, and so each individual vertex of the mesh follows this path. However, for each rendered frame, a new Phoenix mesh must be built from the voxel grid, and so it usually has a different number of vertices than the previous and the next frame. Because of this, individual vertices can not be traced back or forward in time between frames. Instead, motion blur of fluid meshes uses the velocity of vertices which is recorded by the simulation, and moves each vertex back and forward in time along the vertex velocity. This is why the generated liquid mesh does **not** support frame sub-sampling for motion blur. This may cause a mismatch between the liquid and transforming/deforming objects in your scene that interact with it. The fluid mesh is generated from data at the exact rendered frame and fluid data for the preceding or following frames is not used, unlike regular deforming meshes. As a consequence, the liquid and the objects in your scene would synchronize best if those objects do not use additional geometry samples for motion blur.

The Grid Channel Smoothing controls allow you to smooth the Grid Channels loaded from cache files for preview and rendering.

You can smooth the Velocity Grid Channel stored in the simulated caches in case the motion blurred edges are looking jagged.

The recommended values for the smoothest result are all zeroes for the Threshold, Similarity and Random Variation options - this will produce strongest smoothing, evenly applied over the entire Grid, without adding any random variation.



## Displacement

Displacement is a technique intended to add detail to the simulation during the rendering. The idea of the Phoenix displacement is similar to the usual geometry displacement: a texture is sampled, and the corresponding point of the fluid volume or surface is shifted in a direction at a distance determined by the texture. You can plug any V-Ray, 3ds Max or Phoenix texture maps.

You can use the Phoenix Simulator's Mesh Preview option to check how the attached displacement map is affecting the surface when **Mode** is set to **Mesh**, **Ocean Mesh** or **Cap Mesh**.

**Enable** | *displacement* – Enables the displacement effect, and enables the use of a multiplier value to increase or decrease the overall displacement effect.

**Multiplier** | *displmul* – Specifies the displacement amount.

**Map** | *displ2* – Specifies the map used to displace the fluid mesh or Isosurface. Phoenix would automatically detect if a monochrome texture is used and displace the surface along its normals, or if a colored texture is used and treat it as a 3D vector displacement. If a monochrome texture is used, black color with value of 0.0 would mean no displacement, positive colors will extrude the surface outwards, and negative colors will subside the surface. In 3D vector mode, the texture is expected to be in the format used for V-Ray Tangent Vector displacement, where X and Y of the texture are 0.5-based, and the Z direction is 0.0-based. Such a texture is the [Phoenix Ocean Texture](#). For more information on texture mapping in Phoenix, please check the [Texture mapping, moving textures with fire/smoke/liquid, and TexUVW](#) page.

**Vertical Fade Level** | *fadebylevel*, *displfade* – How high above the **Ocean Level** the displacement will stop having effect. This option is available only in **Ocean Mesh** and **Cap Mesh** mode and it is needed for ocean simulations where you have liquid flying or splashing high above the ocean surface, so that the ocean displacement will affect only the calm ocean surface, but will not displace the liquid flying high above the ocean, or you would be able to see small pieces of liquid move up and down as they fly due to the ocean waves displacement. Above the **Vertical Fade Level** there will be no displacement at all, and below it the displacement will be strongest near the **Ocean Level** and will gradually be reduced moving up from the ocean surface. This parameter is a percentage of the grid height, just as the **Ocean Level** option.

**Fade Above Velocity** | *fadebyvel*, *displvelfade* – If the fluid velocity (in voxels/sec) in a voxel is higher than this value, there will be no displacement at all. When the velocity is lower than this value, the higher the velocity, the weaker the displacement will be. This allows you to suppress displacement for the fast moving parts of the fluid where the displacement would visibly disturb the motion in an unnatural manner, and thus you can have only the still ocean surface displaced with waves. This option requires the **Grid Velocity** channel to be exported to the simulation cache files from the [Output](#) rollout.

The Fade Volume feature can be used when the liquid is in contact with a geometry surface such as a shore or a ship and the displacement breaks the contact by moving the liquid mesh away from, or into the geometry.


**Fade Volume** | *usefadeobj*, *fadeobj* – There will be no displacement inside this geometry object, and outside it the displacement will be gradually reduced at a distance specified by the **Volume Fadeout Distance** parameter. This is useful if you have a geometry such as a sea vessel moving in high displaced waves, or a static geometry such as a beach into which the fluid is splashing. In these cases, you can use these geometries as fade volumes and use small **Fadeout Distance** around them, so the displaced liquid mesh and the objects would match precisely - otherwise the displacement may pull away the liquid from such geometries or force the liquid to penetrate them.


**Volume Fadeout Distance** | *displgeomfade* – Specifies the distance in world units around the object where the displacement will fade out going near the object.

## Displacement

☐ Enable 1.0 

Map: No Map

☒ Vertical Fade Level  
20.0 

☐ Fade Above Velocity  
20.0 

☐ Fade Volume  
None

Volume Fade Dist 20.0 