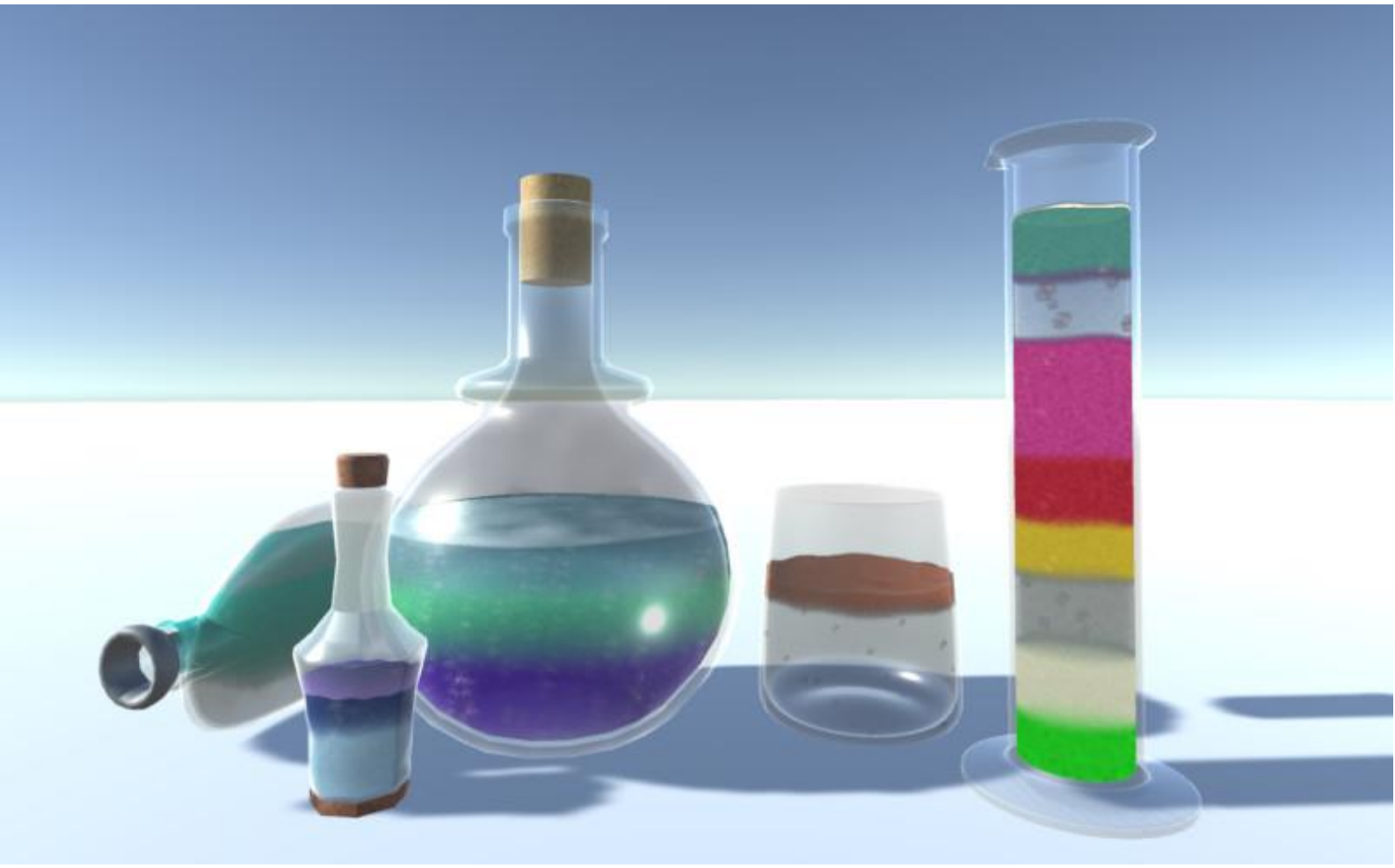


LIQUID VOLUME PRO

QUICK START GUIDE

Kronnect

Contents

Introduction

Thanks for purchasing!

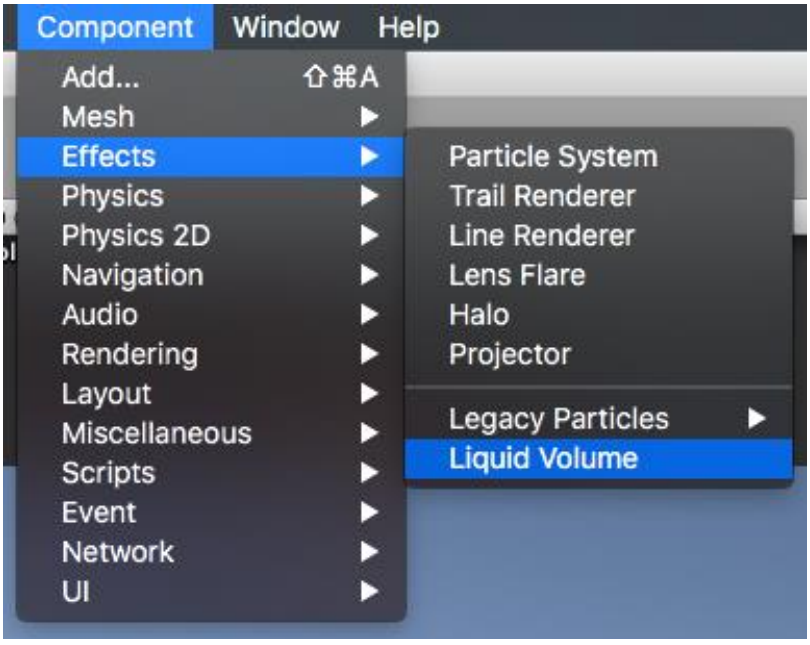
Liquid Volume Pro is a powerful and highly customizable shader that simulates realistic and animated liquid containers. Can be used in real 3D scenes and 2D UI (demos included).

Quick Start

There're several demo scenes included. We highly recommend checking them out before using the asset in your project and to play with the different script values in those scenes. The Potion demo scene contains a Potion prefab which you may also use in your scene. The Beer scene contains a BeerFlask prefab as well.

Add Liquid Volumes to 3D Scenes

To create a liquid volume in a 3D scene, simply attach a LiquidVolume component to any sphere, cylinder or box game object:



For convenience you can find ready to use prefabs located in Resources/Prefabs folder which you may drag and drop to your scene.

Add Liquid Volumes to 2D Scenes (and GUI)

Liquid Volume can also be used to show indicators in your UI. Check demo scene "RPG_LifeAndMagic".

To add a sphere indicador to your scene, locate the CameraLife or CameraMagic prefabs in LiquidVolume/Resources/Prefabs and drop it into your scene. It will create a sphere flask with a custom camera that will render the image to the render texture s LifeRT or MagicRT, located in Resources/Textures.

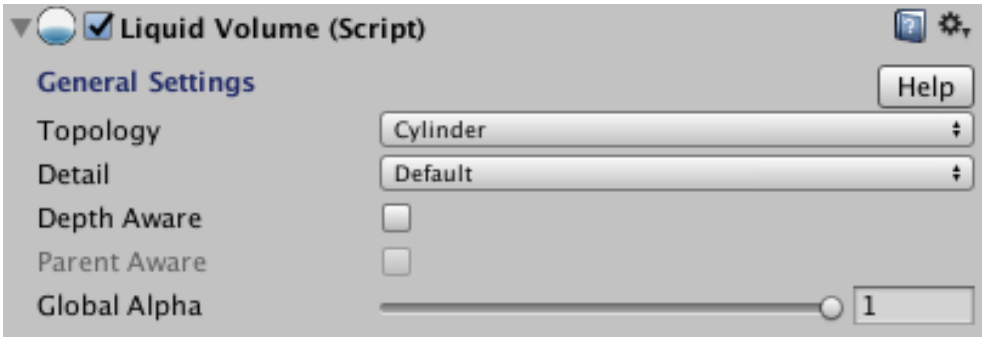
Once you accomplish the step above, you can use the LifeRT texture in a RawImage UI component of your game interface.

The same approach can be used to create other types of indicators.

Inspector Settings

The LiquidVolume component presents many appearance and behaviour options organized in the following sections:

General Settings



- **Topology**: specifies the underline geometry, which must be compatible with sphere, cylinder, cube or irregular types. The potion and beer flask included in the demos uses a cylinder topology for example. However for most non-primitive types choosing Irregular will produce better results at any angle.

- **Detail**: supports one of the following styles:

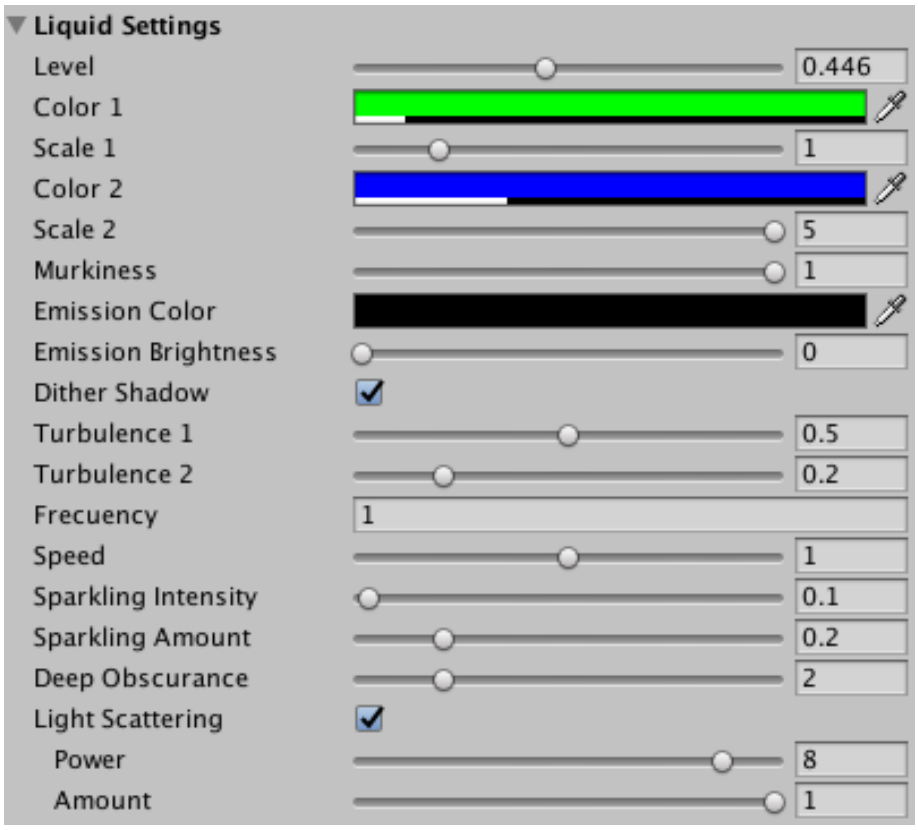
    - Simple (do not uses 3D textures and is compatible with all mobile)
    - Default: includes liquid, foam and smoke effects with a glossy flask effect.
    - Default with no flask (has fewer render passes): same but no flossy flask effect.
    - Texture bump: adds additional texturing and bump mapping options to the flask
    - Reflections: uses a cubemap for reflections.
    - Multiple: supports multiple layers of liquid.
    - Multiple No Flask: same with no flask effect.

- **Depth Aware**: enable this option to allow objects inside the liquid container.

- **Parent Aware**: uses parent object's geometry as a boundary for the current liquid. Use only with irregular topology when liquid volume is inside another object and to prevent seeing through the container other liquids in the background. If you don't see any artifact, don't enable this option as it adds an additional render pass for the liquid containers.

- **Global Alpha**: allows you to control the overall transparency of liquid and smoke.

Liquid Settings



This section controls the appearance of the liquid under the foam.

- **Level**: this is the fill level. Important: if Multiple detail is selected you will be able to customize multiple layers of liquid.

- **Color1, Color2, Scale1 and Scale2:** controls the appearance of the two liquid components which are blended together to produce the final pixel color.

- **Emission, Emission Brightness:** controls the emission color and brightness of the liquid.

- **Murkiness**: amount of noise.

- **Alpha**: global transparency for the liquid, but also for the smoke and foam.

- **Dither Shadow**: enable to apply a dither to the liquid shadow trying to simulate a partially transparent shadow. For best results enable soft shadows in Unity Quality Settings and also in the light setting itself.

- **Turbulence 1:** small turbulence amount.

- **Turbulence 2:** big turbulence amount. Mix wisely with turbulence 1 to produce a reaslistic movement. Note that turbulence also affects smoke and foam.

- **Frecuency:** increase to produce shorter turbulence waves when using models with scale greater than one. A default value of one is fine for most cases though.

- **Sparling Intensity/Amount:** let's you add sparks to the liquid.

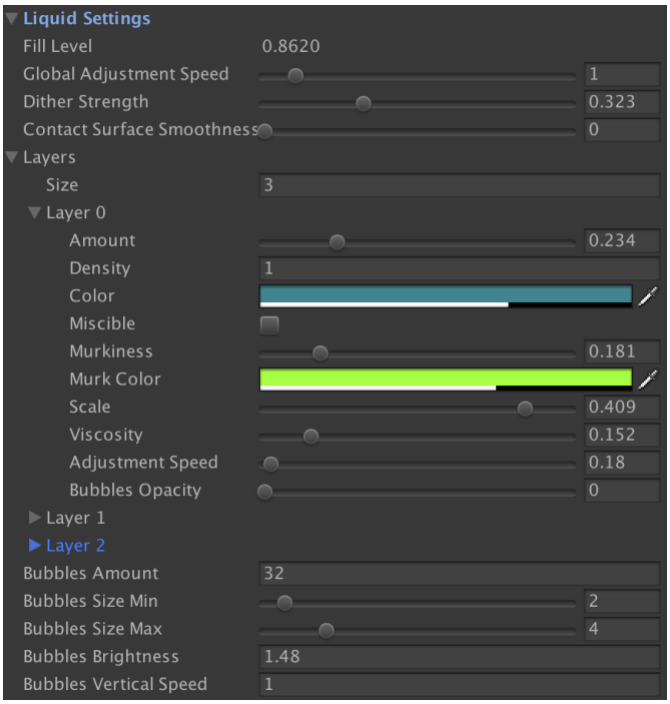
- **Deep Obscurance**: creates a gradient to darker colors at bottom.

- **Light Scattering**: enables backlight passing through the liquid producing a light diffusion effect. Power and Amount parameters controls the focus and intensity of this particular effect.

- **Light Scattering**: enables backlight passing through the liquid producing a light diffusion effect. Power and Amount parameters controls the focus and intensity of this particular effect.

Liquid Settings (only Multiple detail level)

When Detail setting is set to Multiple (or Multiple No Flask) you're presented with many new options. The liquid layers are sorted by Density value and you will be able to define any number of liquid layers as well as its individual properties:



Common settings for all layers:

- **Fill level**: the cumulative fill level for all liquid layers.
- **Global Adjustment Speed**: liquid layers are sorted by density. When density is changed and the layer positions change, this option controls the global speed of change.
- **Dither Strength**: help reducing banding effect between layers.
- **Contact Surface Smoothness**: controls the blending between surface layers.

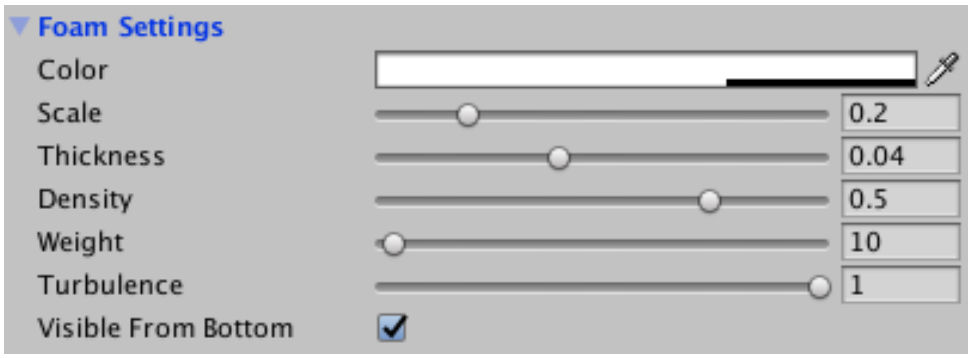
Per layer settings:

- **Amount**: the amount of liquid in this layer (like the Fill level but for each layer).
- **Density**: defaults to 1. Layers are sorted by density. If two layers have the same density and Miscible checkbox is enabled, the two or more layers will mix.
- **Color**: the base color for the layer.
- **Miscible**: enable it on two or more layers to allow layer mixing. Note that only liquids of same density will mix.
- **Murk Color / Murkiness**: color and murkiness intensity.
- **Scale**: scale of the noise used for the murkiness effect.
- **Viscosity**: determines the influence of external forces / turbulences on this layer. Some liquids are stickier than others (ie. honey vs alcohol).
- **Adjustment Speed**: velocity for the liquid to switch layers when density changes.
- **Bubbles Opacity**: the opacity for the bubbles crossing this layer.

Bubbles settings:

- **Bubbles Amount**: amount of generated bubbles.
- **Bubbles Size Min/Max**: range for the bubbles size.
- **Bubbles Brightness**: defaults to 1. Increase to make bubbles shine.
- **Bubbles Vertical Speed**: defaults to 1. Change at will to control bubbles speed.

Foam Settings



This section allows you to customize the foam appearance:

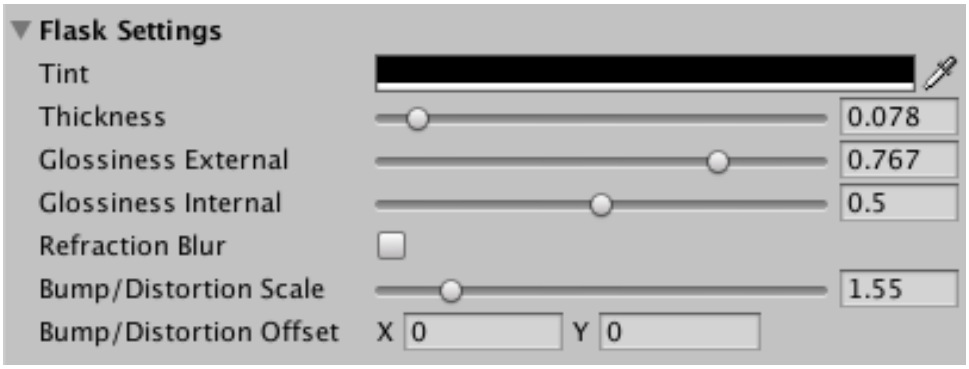
- **Color** and **Scale**: basic color and resolution of the foam effect.
- **Thickness**: relative height of the foam with respect to the flask height.
- **Density**: increase this value to create opaque foam.
- **Weight**: controls the smoothness of the foam at the bottom, near the liquid.
- **Turbulence**: multiplier to the turbulence factors of the liquid. Set this to zero to produce a static foam regardless of the liquid turbulence.
- **Visible From Bottom**: enable this option to allow the foam to be visible from bottom-up, throught the liquid (if liquid alpha is low).

Smoke Settings

This section controls the look and behaviour of smoke effect inside the flask and above the liquid/foam:

- **Color** and **Scale**: as with liquid and foam, these properties controls the basic appearance of the smoke pixels (color and resolution).
- **Speed**: custom speed multiplier for the smoke.
- **Base Obscurance**: makes the smoke darker at the bottom.

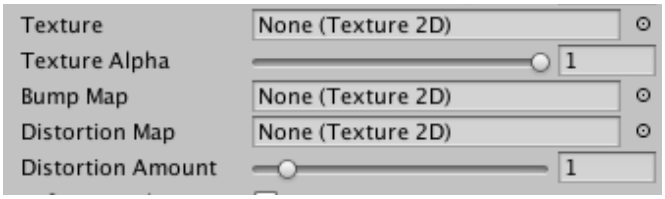
Flask Settings



This sections controls the look of the flask (the container itself):

- **Tint**: color applied to the flask. For transparent containers, reduce the alpha component to zero.
- **Thickness**: flasks have walls, and this property controls the width of the walls.
- **Glossiness**: the “glossiness” or brightness of the external and internal side of the flask walls.
- **Refraction Blur**: when enabled, the visible background is blurred to simulate light refraction.

When “Detail” is set to “Texture Bump” additional options appear in this section:



- **Texture**: allows you to wrap a texture around the flask. This feature supports transparency.
- **Bump Map**: assign a normal map texture to create a relief effect.
- **Distortion Map:** assign a texture (like normal map) which will be used to compute additional image distortion.

Physics Settings

This section controls the response to quick changes in position or accelerations:

- **React to Forces**: when enabled, the turbulence of the liquid will react to external forces or accelerations. The sphere topology allows rotation of liquid inside the container as well as change in turbulence. Cylinder and Cube topologies will only alter the turbulence.
- **Mass**: defines the “mass” of the liquid. A higher value will increase the weight of the liquid thus it will move less.
- **Angular Damp**: defines the internal friction of the liquid. A higher value will make the liquid return to calm state faster.
- **Ignore Gravity**: enable this option to force liquid to rotate with the flask. Note that “React To Forces” option disables this.

Advanced Settings

This sections controls “rare” parameters that you usually won’t need to use but are useful as well:

- **Smoke/Liquid/Foam Ray Steps**: the shaders uses a ray-marching approach to compute the pixel color. These settings specify the number of samples taken per each ray traced. The more samples, more dense and quality will be the liquid, but also slower.

- **Noise Variation**: the asset includes three precomputed 3D noise textures. Move this slider to switch between them. Experiment with the different possibilities to get the best result.

- **Upper Limit**: when using models that are not symmetrical (like a bottle which has a cork which you want to top the liquid effect) you can use this property to limit the height to which the liquid can reach.

- **Extents Scale**: optional multiplier to the real model extents (size). You can use this property to reduce the area of the geometry used by the liquid volume.

- **Allow View From Inside**: if enabled, the asset will detect if the camera enters the liquid container and will render appropiately. Note that this is an experimental option and some other properties are not supported when the camera is inside the container, like turbulence.

- **Bake Current Transform**: many times when you import a model from a third party application, either rotation or scale won't match Unity standards. You usually compensate this applying some transform but the problem is the mesh itself is incorrectly oriented, so some calculations done by Liquid Volume shader will not work. To fix this, you can bake the transform of the gameobject to the mesh itself using this option. After executing this operation, gameobject's rotation will be reset and scale will be set to 1.

- **Center Pivot**: similar to the previous issue, some times you may want to correct or move the pivot point of the mesh to the center. This operation does just that, effectively centering the mesh around a common center.

- **Render Queue**: by default, Liquid Volume renders at Transparent+1 in the render queue (which is 3000 + 1). You can change it to 3000 so it renders as a normal gameobject or use any other value if needed. Changing this value has the same effect that editing the shader and changing the RenderQueue tag.

- **Double Sided Bias**: when using Irregular topology and open flasks, you may want to use this parameter to ensure only the far and face viewing triangles are being rendered. Problem with open flasks is that their boundaries has two sides so usually there's some overdraw. Using this parameter ensures the only one part of the geometry is being rendered. Try different values until you get the result you need.

- **Rotation Level Bias**. Liquid might seem to disappear under certain rotations. This effect is more visible when the level of liquid is low and the flask is not 100% straight. If you see liquid disappear when rotating the flask, try increasing this value (probably until 1.0).

- **Legacy Rendering**. Enable this option to activate shader paths that are compatible with older devices. Note that enabling or disabling this option will recompile all shaders and will take some time.

Shader Features

You can disable some shader features completely to make the execution a bit faster or increase compatibility on older devices. Disabling these settings on modern devices or PC won't improve performance. Only disable Floating Point buffers if Liquid Volume does not render correctly.

Note that when a shader feature is disabled, the shader code is not compiled, hence it can't be toggled on/off at runtime. If you need some features on or off depending on the situation, then leave these shader features enabled.

- **Light Scattering**: disables shader code related to diffusion of directional light through the liquid.

- **Smoke**: disables ray-marching code through the liquid container to generate smoke (so only liquid/foam is rendered). This is similar to unchecking the "Visible" option of Smoke section but disabling smoke will still compile the shader variants that include the smoke rendering.

- **Bubbles:** disables bubble-related code from the shader. If you're not going to use bubbles, it's more performant to remove the shader code by disabling this shader feature than simply deactivating it from the Bubbles section.

- **Floating Point Buffers**: disable only if Liquid Volume does not render correctly on older devices that do not support floating point render textures. Disabling this option will make the shaders fallback to use low precision buffers which will render with less precision.

- **Orthographic Projection:** enable this option to support orthographic camera.

API (using the Liquid Volume with C#)

All the script properties are accesible through its class. To change any property using C# you need to get a reference to the LiquidVolume component of the target gameObject using:

```
using LiquidVolumeFX;
…
LiquidVolume liquid = <gameobject>.GetComponent<LiquidVolume>();
```

Then, to change the current fill level to 65%, you can do:

```
liquid.level = 0.65;
```

If detail level is set to Multiple, the individual layer levels can be changed through the **liquidLayers** array and then calling:

```
liquid.UpdateLayers(false or true); // pass “true” to force immediate update
```

In addition to the look & feel properties, you will find useful the following API:

`float liquid.liquidSurfaceYPosition;`
Returns the world space vertical position of the liquid surface (without turbulences).

`public void MoveToLiquidSurface(Transform transform, BuoyancyEffect effect, Transform root = null, float dampen = 0.8f)`
Moves an object to liquid surface. This method can approximate the displacement and rotation caused by turbulences. Check demo scene “Buoyancy” for an example.

- transform: the transform of the object to be moved.
- effect: kind of buoyancy effect, if only position is changed or position and rotation as well.
- root: if effect includes rotation, a root or parent object to the object to be moved must be provided.
- dampen: amount of turbulence received by the object (0-1).

`public bool GetSpillPoint(out Vector3 spillPosition, float apertureStart = 1f)`
Calculates approximate point where liquid starts pouring over the flask when it’s rotated. Returns true if liquid spills out.

- spillPosition: the position where the liquid starts pouring.
- apertureStart: a value that determines where the aperture of the flask starts (0-1 where 0 is the flask center and 1 is the very top of the mesh).

`public bool GetSpillPoint(out Vector3 spillPosition, out float spillAmount, float apertureStart = 1f, LEVEL_COMPENSATION rotationCompensation = LEVEL_COMPENSATION.None)`
Calculates approximate point where liquid starts pouring over the flask when it’s rotated. Returns true if liquid spills out.

- spillPosition: the position where the liquid starts pouring.
- spillAmount: a value that represents an amount of liquid spilt.
- apertureStart: a value that determines where the aperture of the flask starts (0-1 where 0 is the flask center and 1 is the very top of the mesh).
- rotationCompensation: the method to compute the amount of liquid spilt.

`public void BakeRotation();`
Applies current transform rotation and scale to the vertices and resets the transform rotation and scale to default values. This operation makes the game object transform point upright as normal game objects and is required for Liquid Volume to work on imported models that comes with a rotation

`public void CenterPivot(Vector3 offset, bool closeMesh);`
This operation computes the geometric center of all vertices and displaces them so the pivot is centered in the model:
- offset: displacement added to the pivot.
- closeMesh: if true, the mesh will be modified so it results in a closed shape (useful for open glasses for example).

`public void SetLayerDensity(int layerIndex, float newDensity, bool keepVolume);`
Convenient function that can change the density of a layer while also keeping the same volume. Usually increasing the density of a liquid will decrease the volume (amount must be increased to compensate; this method does that):
- layerIndex: layer to be modified.
- newDensity: the new density value.
- keepVolume: the amount of the liquid in this layer will be modified so it keeps the previous volume.

`public float GetMeshVolumeWS()`
Returns the approximate volume of a mesh.

`public float GetMeshVolumeUnderLevel(float level01, float yExtent)`
Approximates the mesh volume under a fill level expressed in 0-1 range:
- level01: fill level (0-1)
- yExtent: y-extent of the volume (ie. renderer.bounds.extents.y).

`public float GetMeshVolumeWSFast()`
Returns the approximate volume of a mesh (fast method).

`public float GetMeshVolumeUnderLevelWSFast(float level)`
Returns the approximate volume of a mesh (fast method):
- level: fill level (0-1)

FAQ (Frequent Asked Questions)

I have a custom flask which is not primitive-type, how can I make the liquid fill it properly?

For non-primitive flask types (non cubic, cylindrical or spherical flasks), choose “Irregular” topology. Also if the model pivot is not at its center (for instance, the 0,0,0 position sits at the bottom of the flask instead of the middle) you may want to click “Center Pivot” which will modify the mesh vertices so the pivot gets reseted to the center of the model. This is necessary so the level property and other effects can work properly. Otherwise you can try modifying the Extents Scale or UpperLimit to try to fit the liquid to the extents of the flask without moving the pivot.

I need to use a non-capped (non closed) cup or flask, how can I use that?

See what happens when using an open container in the image below (cylinder topology on the left, irregular topology on the right):

The Cylinder topology assumes the container is closed, so it will automatically work as if the glass was closed on top. In this case a Cylinder topology could work better than Irregular topology.
In the second glass, the Irregular topology uses the mesh itself to define the initial and end position of each volumetric ray. Since there’s no mesh on top of glass, the ray on the top edge starts in the interior of the glass instead of the top, producing the clipping in the image.

Another common solution to non-capped containers consists of putting a liquid volume with a cylinder topology inside your cup or open container. Set the detail setting to “Default No Flask” which will render just the liquid and it will work. Check out demo scene “NonCapped” for an example.

Does it work on mobile?

Liquid Mobile has been tested with success on Samsung Galaxy 6 and iPhone 7.
Note that some options add some extra overhead so you may want to toggle them off:

- Refraction Blur: when you enable this option, a CommandBuffer is attached to the main camera and a full screen blur is performed to produce the effect. Obviously this can add some performance penalty in mobile.
- Irregular Topology: this is the most complex topology for the shader. It involves another CommandBuffer which renders the geometry back faces to perform the ray-tracing. If you need more performance you may want to try the Cylinder topology instead.
- Depth Aware: enabling this option will make the fragment shader to check the Camera Depth Buffer. Disabling this option will save a texture fetch.

On mobile, I get some warnings in the console like "Tiled GPU Perf…"?

This is a known issue related to the use of CommandBuffers on mobile. You can ignore this message as in this case it does not affect the execution of the shader.

How can I obtain the Y position of the surface level?

Use the property liquidSurfaceYPosition (see previous section API).

Liquid does not get illuminated by additional lights

To support multiple lights, edit the shader corresponding to the detail level and look for/remove the word "noforwardadd".

Compilation takes lot of time!

To reduce compilation time, you can remove unused Liquid Volume shaders located on the Liquid Volume/Resources/Shaders folder:

- LiquidVolumeDefault.shader: default style.
- LiquidVolumeDefaultNoFlask.shader: default with no flask style.
- LiquidVolumeBump.shader: texture/bump style.
- LiquidVolumeReflections: reflections style.
- LiquidVolumeSimple: simple style.
- LiquidVolumeSmoke: only smoke style.
- LiquidVolumeMultiple: allows several layers of liquid.
- LiquidVolumeMultipleNoFlask: several layers without flask.

in addition to remove unnecessary shaders, you can edit the shaders and reduce some features. Look for these lines in the shader files:

#pragma multi_compile __ LIQUID_VOLUME_DEPTH_AWARE
Comment out the above line if you won't use Depth Aware option in your build.

#pragma multi_compile __ LIQUID_VOLUME_DEPTH_AWARE_PASS
Comment out the above line if you won't use Parent Aware option in your build.

Expand the Shader Features section in Liquid Volume section. You'll find additional options to fine tune the shader and locate the shades in your project.

Liquid does not render correctly when platform is set to WebGL

Select Edit -> Graphic Emulation -> WebGL 2.0 in Unity Editor. When building the app, the asset will be compatible with WebGL 1.0 but in Unity Editor it requires emulation with WebGL 2.0.

Support

For additional support, please visit kronnect.com forum.
On kronnect.com you'll find latest beta updates for this asset and many others!