



Liquid Shader (3D)

v1.0
Documentation

Contents

1	Introduction	2
2	Documentation	3
2.1	Universal Render Pipeline (URP)	3
2.1.1	Universal Renderer	3
3	Support	15

Section 1

Introduction

The Liquid shader (3D) turns a spline into a living body of liquid – lava, water, blood, honey, acid, slime, mud, oil, molten metal, poison, radioactive waste, and anything in between. The liquid surface flows, rolls in waves, breaks into ripples, and grows floating patches of crust, foam, or scum, all of which are fully customizable.

WARNING

This shader supports only Universal Render Pipeline (URP) on Unity 6.0+. Built-in Render Pipeline (BIR) and High-Definition Render Pipeline (HDRP) are not supported.

Two helper scripts are included. The **Liquid Surface** component builds a dense, evenly tessellated surface mesh from a spline – a closed spline becomes a lake or a pool, an open one a river – so that the wave and ripple displacement has enough vertices to work with. The **Quick Liquid Setup** component condenses the many shader properties into five intuitive controls, letting you dial in a convincing liquid in seconds.

Thirteen ready-made liquids ship with the asset – each with its own textures and material – along with a demo scene for every one of them. Feel free to check them out for inspiration, or simply use them as they are.

Section 2

Documentation

2.1 Universal Render Pipeline (URP)

2.1.1 Universal Renderer

The Universal Renderer is the default URP Renderer when creating a 3D project.

Setup

The following paragraph presents the steps necessary to make the Liquid (3D) shader work in your project. Visit the following link for image-aided instructions:

<https://docs.project-gamedev.com/liquid-shader-3d/setup.html>

The Liquid (3D) shader is, in principle, the same as the regular URP Lit shader you would use in your Unity project, only reconstructed and expanded. Since the liquid surface is animated by displacing the vertices of its mesh, the shader is meant to be applied to a mesh generated by the **Liquid Surface** component:

- Create a new empty GameObject in your scene (**GameObject** → **Create Empty**), and add the Liquid Surface component to it (**Add Component** → **Project GAMEDEV** → **Essential 3D Shaders** → **Liquid Surface**). The **Spline Container**, **Mesh Filter**, and **Mesh Renderer** it needs are added automatically, and the surface starts out as a small, round lake.
- Shape the outline of your body of liquid by editing the spline in the Scene view, with the spline tools of the Splines package. A closed spline is filled (a lake, a pool, a lava basin), while an open spline becomes a strip (a river, a stream, a lava flow). Consult with the Liquid Surface section for more information.
- Create a new material for your liquid; then, select **Project G.A.M.E.D.E.V./Essential 3D Shaders/Liquid Shader (3D)** as the shader source for this material.
- Assign the new material to the **Mesh Renderer**.

That's it! The shader is now active. Feel free to play around with the material properties to find a liquid that you are happy with, start from one of the included materials, or

attach the `QuickLiquidSetup.cs` component described in the Quick Liquid Setup section to shape the liquid with a handful of controls instead.

Keep in mind that certain post-processing settings (most notably **Bloom** and **Tonemapping**) will have a profound effect on the emissive parts of your liquid. The demo scenes ship with the Post-Processing Profiles used to author them, in the `/Post-Processing Profiles` folder.

WARNING

The Liquid Surface component requires Unity's **Splines** package. Make sure it is installed in your project (**Package Manager** → **Unity Registry** → **Splines**) – without it, the included scripts will not compile.

NOTE

The waves and the ripples are produced by displacing the mesh vertices of the liquid surface. If the surface looks blocky or barely moves at all, raise the **Mesh Detail** of the Liquid Surface component. Plain meshes, such as Unity's built-in Plane, are not supported: they lack the edge mask that the shader reads from the vertex colors, as well as the vertex density that the waves need.

Shader Options

The material inspector groups the shader options into six categories – **Main Surface**, **Emission**, **Liquid**, **Waves**, **Ripples**, and **Patches** – described below in the same order.

Main Texture The main (color) texture of the liquid. This is the texture that is scrolled across the surface by the **Liquid** options. Consult with the Advanced section on how it is rendered.

Albedo An additional color multiplied with the one already present in the main texture. Use this to further customize the look of your liquid. The Albedo color is also the lit color of the surface underneath the main texture – consult with the Advanced section for more information.

Alpha Cutoff This is a threshold value – if the alpha value of the **Albedo** color is less than the alpha cutoff, the liquid will not be rendered. The alpha channel of the main texture does not affect it.

Ambient Occlusion A value between 0 and 1 that controls how much ambient (indirect) light reaches the surface. At **1**, the surface receives the full ambient lighting; lower values darken it, as if the surface were occluded. Unlike in the standard URP Lit shader, this is a single value for the whole material rather than a texture.

Metallic Map The standard metallic map option in the default URP Lit shader. The metallic map is a black-and-white texture that determines the surface type of the object. It is used to make certain parts of the liquid reflect light like a metal would (metallic), and others like sand would (smooth). The shader reads the metallic value from the texture's red channel, so disable **sRGB (Color Texture)** in the texture's import settings. If no metallic map is assigned, the entire surface is treated as fully metallic.

Metallic This value determines how “metallic” the surface should be, i.e., whether it should reflect light like a metal or a wet surface, or should absorb it like sand or another rough, non-metallic surface.

Normal Map The standard normal map option in the default URP Lit shader. The normal map is used to simulate finer surface detail by containing information on how the light should be reflected by the surface. Make sure the texture's **Texture Type** is set to **Normal map** in its import settings. Every included liquid ships with a matching normal map.

Normal Strength This value is multiplied with the normal map and is used to make the effect stronger or weaker. A value of **1** applies the normal map at its original strength, and **0** disables it. A negative value will mirror the 3D-effect (i.e., protruding surfaces will now be concave, and vice versa).

Vertical Offset Shifts the whole liquid surface along its normals – i.e., up or down, for a level surface – by the given distance in local units. Use this to sink the liquid deeper into its basin or to make it swell above the rim, without editing the spline itself.

Emission Color The emission color as an HDR color. Keep in mind that **different post-processing effects (most notably bloom)** will have a **profound** effect on the look of your emission. This is the property that makes lava, magic liquid, and radioactive waste glow.

Emission Map The standard emission map option in the default URP Lit shader. A black-and-white map that determines which parts of the liquid will emit light. The white pixels of the texture are areas where the emission will have an effect, and the black pixels are the areas with no emission. Like the main texture, the emission map flows across the surface with the **Liquid** options.

Emission Extra Intensity Since the default HDR color has a limited intensity (-10 to 10), this shader offers an additional intensity option that allows you to make your emission even stronger. A value of **1** leaves the emission unchanged, values above 1 strengthen it, and values below 1 weaken it. The intensity grows exponentially, so small steps (e.g., from 1.1 to 1.5) already make a clear difference. New materials start at **0**, at which the emission only shows where the liquid is seen at a grazing angle – set it to 1 for a regular emission. Consult with the Advanced section on how this option interacts with the viewing angle.

NOTE

The emission – including the main texture, see the Advanced section – is additionally faded out towards the outline of the liquid surface, using the edge mask written by the Liquid Surface component. This is what gives lava its cooled, darkened rim where it meets the rock. The width and the falloff of that rim are controlled by the **Edge Mask Width** and **Edge Mask Softness** settings described in the Liquid Surface section.

Liquid Movement Speed How quickly the liquid textures flow across the surface. Set to 0 to bring the flow to a complete standstill. Larger values produce a faster current.

Liquid Movement Direction The direction of the flow, in degrees (0 – 359). 0 flows along the +X axis, 90 flows along the +Z axis.

Wave Speed How quickly the waves travel across the surface. Set to 0 to freeze the waves in a single, static deformation.

Wave Direction The direction in which the waves travel, in degrees (0 – 359). Off-setting this slightly from the **Liquid Movement Direction** makes the surface look considerably more natural, since the waves then no longer run exactly along the current.

Wave Frequency How closely the wave crests follow each other (0 – 50). Low values produce long, rolling swells; high values produce short, choppy waves.

Wave Amplitude How far the vertices are displaced by the waves, i.e., how tall the waves are. Set to 0 to flatten the surface completely.

Wave Distortion Strength The amount of noise mixed into the waves (0 – 10). At 0 the waves are perfectly regular. Raising this value varies the height of the waves across the surface, so that no two crests look quite the same.

Ripple Speed How quickly the ripples – the second, finer layer of displacement on top of the waves – animate.

Ripple Size The size of an individual ripple (0 – 1). Small values produce a dense pattern of tiny ripples; large values produce a few broad ones.

Ripple Amplitude How far the vertices are displaced by the ripples. Set to 0 to disable the ripple layer entirely and leave only the waves.

Ripple Frequency How far out of step neighbouring parts of the surface ripple. Larger values make the fine surface detail look busier and more agitated, so that it seems to boil rather than gently undulate.

NOTE

The waves and the ripples are two independent displacement layers, and they are meant to be used together. A thin liquid such as water usually wants fast, small, plentiful ripples over shallow waves, whereas a thick liquid such as lava or honey wants tall, slow waves and barely any ripple at all.

Patch Speed How quickly the patches – the darker crust, foam, scum, or mud floating on the surface – drift across the liquid.

Patch Density How many patches are spread across the surface (0 – 10). Low values produce a few large islands; high values produce many small ones.

Patch Movement Direction The direction in which the patches drift, in degrees (0 – 359). It is usually worth setting this a little apart from the **Liquid Movement Direction**, so that the patches appear to be carried by the current rather than glued to it.

Patch Threshold The cut-off that decides how much of the surface is covered by patches (0 – 2). Low values let the patches spread out and merge into a closed crust; high values leave only their cores visible.

Patch Softness How sharply the patches end (0 – 2). At 0 the patches have hard, clearly defined borders; larger values let them fade gradually into the surrounding liquid.

Patch Color The color of the patches. Dark, desaturated colors read as cooled crust or scum, while lighter ones read as foam. The patches are drawn with this color both on the lit surface and in the emission, so they keep their color regardless of the lighting.

Patch Distortion Strength The amount of noise warping the outline of the patches. At 0 the patches keep the smooth, cell-like shape they are generated with. Larger values eat into their borders and make them look torn and irregular.

Patch Transparency How much of the liquid underneath shows through the patches (0 – 1). At 0 the patches fully replace both the color and the emission of the liquid below them; at 1 they are blended over the liquid at half strength.

i NOTE

To remove the patches completely, raise **Patch Threshold** to its maximum of 2 and lower **Patch Softness** to 0.

API

Visit <https://docs.project-gamedev.com/liquid-shader-3d/api.html> for the Liquid Shader (3D) API documentation. This API allows you to modify all shader values in-code, from your own scripts, if necessary.

Advanced

i NOTE

The **Emission Extra Intensity** option is weighted by the viewing angle, and applies to the emission map only (not to the main texture). Surfaces facing the camera receive the full effect, while surfaces seen at a grazing angle remain almost unchanged. As a result, values above 1 make the emission brightest where the surface faces the camera, while values below 1 fade the camera-facing emission first and keep it at grazing angles – at 0, the emission only shows where the liquid is seen nearly edge-on.

i NOTE

With positive amplitudes, the waves and the ripples only ever raise the surface above its resting height – never below it. If the liquid swells over the rim of its basin, lower it with a negative **Vertical Offset**. Since the displacement happens on the GPU, the colliders fitted by the Liquid Surface component follow the resting surface rather than the waves.

i NOTE

The waves and the ripples are computed in the local space of the GameObject, along the normals of the surface. Moving, rotating, or scaling the GameObject therefore moves, rotates, or scales the wave pattern along with it – scaling a liquid surface up makes its waves taller as well.

Liquid Surface

The Liquid shader animates the liquid by displacing the vertices of its mesh, so it needs a mesh that is dense and evenly tessellated. It also needs to know where the liquid meets the shore, in order to fade out the emission there.

`LiquidSurface.cs` provides both. It turns the first spline of the **Spline Container** on the same `GameObject` into a liquid surface mesh: it fills the spline with an evenly tessellated grid of triangles, generates the UVs for it, keeps optional colliders in sync, and writes an **edge mask** into the red channel of the vertex colors, which the shader uses to fade the emission out towards the outline of the surface.

The surface faces the local +Y axis, so draw the spline in the local XZ plane, as seen from above. How the mesh is built depends on whether the spline is closed or open:

- A **closed** spline is filled. The interior blends smoothly between the heights of the outline, so a level outline produces a flat surface, and an outline that rises on one side produces a tilted one.
- An **open** spline becomes a strip of **Open Spline Width**, centered on the spline. The strip follows the height of the spline along its length and stays level across its width – ideal for a river running downhill.

NOTE

To raise or lower the liquid without editing the spline, use the **Vertical Offset** of the material, described in the Shader Options section, or move the `GameObject` up or down. The material's offset is applied on the GPU, so it shifts every surface that shares the material, and the colliders stay at the height of the spline. Moving the `GameObject` takes the colliders along.

Add the component via **Add Component** → **Project GAMEDEV** → **Essential 3D Shaders** → **Liquid Surface**. It exposes the following settings:

Mesh Detail The density of the generated mesh (1 – 128), i.e., the number of vertex columns and rows per local unit. The cells of the mesh are kept square, so the waves and the ripples keep their shape whichever direction they travel in. This is the single most important setting of the component: higher values produce smoother, more detailed vertex displacement, at the cost of a heavier mesh. Small pools can afford the maximum, while large lakes and seas need far lower values – see the warning below.

Open Spline Width The total width of the generated strip when the spline is **open** rather than closed. The spline stays centered inside the strip. This setting is ignored for closed splines, which are simply filled.

UV Mode How the textures are projected onto the surface. The projection is always planar, from above. Three modes are available:

- **Local Tiling** (default) – the textures repeat every **Tile Size** units in the local space of the `GameObject`, so they move, rotate, and scale together with it.

- **World Tiling** – the textures repeat every **Tile Size** units in world space, so that neighbouring liquid surfaces line up seamlessly. The mesh is rebuilt whenever the `GameObject` moves, rotates, or scales.
- **Stretch To Bounds** – a single copy of the textures is stretched across the bounds of the surface, repeated **Stretch Tiling** times.

Tile Size The number of units per texture repeat in the **Local Tiling** and **World Tiling** modes. Larger values make the liquid textures appear larger.

Stretch Tiling The number of texture repeats across the bounds of the surface in the **Stretch To Bounds** mode.

Edge Mask Width The horizontal distance from the outline of the surface over which the edge mask fades out. The mask is stored in the red channel of the vertex colors and is used by the shader to darken the emission near the outline – a wide value produces a broad, cooled rim, while a very small value effectively disables the rim.

Edge Mask Softness The softness of that falloff (0 – 1). At 0 the mask ends abruptly, giving a hard border; at 1 it fades smoothly across the full edge mask width.

Bounds Padding The distance added around the bounds of the mesh on every side. Unity culls a mesh based on its bounds, which do not account for the waves and the ripples, so without padding the displaced parts of the liquid could disappear at the edges of the screen. The waves and the ripples raise the surface by at most a tenth of **Wave Amplitude** plus **Ripple Amplitude**, in local units; add the **Vertical Offset** on top of that. The default of 2 comfortably covers all of the included liquids.

Update Colliders When enabled, a **Mesh Collider** on the same `GameObject` receives the generated mesh, and a **Box Collider** on the same `GameObject` is fitted to the surface, extended by **Bounds Padding** above and below. Use them, for example, as a trigger that detects the player falling into lava. The component does not add the colliders itself – add the ones you need.

NOTE

Liquid Surface rebuilds its mesh automatically, both in the Editor and in Play Mode, whenever the spline, the settings, or the attached colliders change. The mesh is not saved with your scene – it is regenerated whenever the scene loads, so it does not bloat your scene files.

All settings are also available in code. Each one is exposed as a property of the same name – for example, `MeshDetail`, `EdgeMaskWidth`, or `UpdateColliders` – with the exception of **UV Mode**, which is exposed as `UVProjection`. Changes are picked up on the next `Update`; call `Rebuild()` to apply them immediately.

⚠ WARNING

Each liquid surface is limited to 65,535 vertices and indices, so that its mesh can keep the compact 16-bit index format. The component respects that limit and reduces the effective detail of very large surfaces rather than fail. If a huge surface looks coarser than a small one at the same **Mesh Detail**, this is why – split it into several smaller liquid surfaces instead.

i NOTE

A Liquid Surface uses only the first spline of its Spline Container. To create several separate bodies of liquid, give each one its own `GameObject` with a Liquid Surface. Holes, such as islands, cannot be cut into a surface – let the terrain or other geometry rise above the liquid instead.

i NOTE

Keep the bends of an open spline wider than its strip. Tight corners are cleaned up automatically, but where a river passes closer to itself than the width of its strip, the strip will overlap itself.

Quick Liquid Setup

The Liquid shader exposes a lot of properties, and finding a believable combination of all of them by hand takes time. `QuickLiquidSetup.cs` exists to skip that step: it maps five intuitive controls onto the whole set of shader properties, so that a plausible liquid is only a couple of sliders away.

Attach the component to the same `GameObject` as your Liquid Surface via **Add Component** → **Project GAMEDEV** → **Essential 3D Shaders** → **Quick Liquid Setup**. It updates immediately, both in the Editor and in Play Mode, and offers the following settings:

Movement Speed The base speed of the whole liquid – the flow, the waves, the ripples, and the patches. Each of those layers receives its own multiplier on top of this value, so that they never move in lockstep. Set to 0 to bring the liquid to a complete standstill.

Movement Direction The primary direction of movement, in degrees (0 – 360). 0 moves along the local $+X$ axis, 90 moves along the local $+Z$ axis. The waves and the patches are automatically offset by a few degrees from this direction to keep the surface from looking mechanical.

Viscosity How thick the liquid is (0 – 1). Thin liquids such as water move quickly, with fine, lively surface detail. Thick liquids such as lava, honey, or tar crawl, with a heavier, slower surface response and a more pronounced relief from the normal map.

Volatility How energetic the surface is (0 – 1). Low values produce an almost still surface; high values produce tall waves, strong ripples, and pronounced vertex displacement.

Patch Preset The behaviour of the patches floating on the liquid, as one of seven presets: **None**, **Subtle**, **Sparse**, **Balanced**, **Dense**, **Crust**, and **Slick**. **Crust** covers the surface in large, hard-edged plates of cooled crust; **Slick** produces a thin, oily film; **None** switches off the distortion of the patches and keeps only a faint trace of them. The color of the patches is not part of the preset – set it on the material, with the **Patch Color** property.

NOTE

The component writes its values through a `MaterialPropertyBlock`, which means it never modifies your material asset. Disabling or removing the component restores the material to exactly the values you authored in the inspector, and two liquid surfaces sharing one material can still be given different Quick Liquid Setup settings.

All five settings are also available in code, as the `MovementSpeed`, `MovementDirection`, `Viscosity`, `Volatility`, and `Patches` properties. Assigning to any of them re-applies the setup immediately, which makes it straightforward to have, for example, lava thicken and slow down as it cools. The **Apply Quick Liquid Setup** context-menu entry on the component does the same thing on demand.

i NOTE

Because **Quick Liquid Setup** overrides the movement, wave, ripple, and patch properties of the material – as well as its **Normal Strength** – changing those properties on the material itself will have no visible effect while the component is enabled. Remove or disable the component if you would rather author every property by hand.

⚠ WARNING

Quick Liquid Setup requires a **Renderer** on the same **GameObject**, and only one instance of the component per **GameObject** is allowed. If nothing happens after attaching it, make sure it sits on the same **GameObject** as your Liquid Surface, and that its Mesh Renderer has a Liquid Shader (3D) material assigned to it.

Section 3

Support

In case you experience any problems with the shader, can't manage to set it up properly, or would simply like to propose an improvement, don't hesitate to contact us at contact@project-gamedev.com, or fill out the form at <https://project-gamedev.com/contact.html>.

If you are satisfied with this product, please consider leaving a review on the Unity Asset Store page (<https://assetstore.unity.com/packages/slug/413772>).

Best of luck with your game development journey!