



Drop-In Water — Realistic URP System

Documentation · v1.0.0 · Havoila Studio

Havoila Studio · Version 1.0.0 · Unity 6000.3 and newer · Universal Render Pipeline 17 and newer

Drop-In Water brings oceans, lakes and rivers to Unity's Universal Render Pipeline. The waves are simulated with FFT from the wind, the swell and the depth, and the water is shaded from how light is absorbed and scattered in it. Floating objects feel the same waves the camera sees, and boats and splashes push the water with rings and wakes.

There is no setup: add an **Ocean** to a URP scene and press Play. There is no renderer feature to add and nothing to change in the URP asset. Gameplay code can ask for the height, the slope and the motion of the water anywhere. The answer comes immediately, costs microseconds, and is the same on every machine, so a multiplayer game only has to share its clock.

1. Key features

- **FFT ocean.** Wind waves come from the wind speed, direction and fetch (JONSWAP spectrum). A separate swell comes from distant weather. Four cascades cover every scale, from 1 km swells to centimetre ripples, with sharp crests and flat troughs.
- **Infinite surface.** A camera-centred level-of-detail mesh reaches the horizon, with no seams and no popping.
- **Lakes and rivers.** Lakes are wave water inside a box, at any height. The wind raises waves of the size a real lake of that length would get. Rivers follow a spline and flow downhill: they run faster on slopes and slower at the banks, and turn white in rapids. A river blends into the lake or sea it flows into or out of: it joins their level and fades into their water, whose waves calm under its current.
- **Shorelines.** Bake the terrain once. Waves then shrink and break into surf in shallow water, foam gathers along the shore, and dry land below sea level stays dry. Each wave runs up the beach with a small breaking bore and spray, then drains back, pulling its foam into streaks and leaving the sand wet and shiny.
- **Physically based shading.** The water's colour comes from depth-based absorption and scattering, and light shines through wave crests. The surface refracts and reflects the scene (screen-space reflections, reflection probes and the sky). The sun makes glints on the water and caustics on the seabed. Whitecaps appear where the waves break.
- **Underwater.** Fog and colour loss, a waterline across the lens, sun shafts shaped by shadows and caustics, and a seamless crossing through the surface.
- **Buoyancy. Floating Body** applies Archimedes' force and water drag; automated tests check the floating height against the exact Archimedes draft. Objects drift with the waves and down rivers. All floating bodies are computed in one Burst job per physics step.
- **Interaction waves.** Objects make rings and bow waves, and moving boats leave a true Kelvin wake. Every wavelength travels at its real speed (exact spectral solver), and foam appears where the water is churned.
- **Gameplay queries.** Height, normal and velocity of the water at any point, from a CPU copy of the waves. Queries are safe in FixedUpdate, run thousands at once in Burst jobs, and also work on a server with no GPU.
- **Multiplayer ready.** The waves are a function of the settings, the seed and the time. Every client computes the same sea.
- **Editor tools.** Every Havoila Studio feature is in one **Tools** menu. Inspectors show the material inline, river handles work in the Scene view, and debug views help tune the look.

2. Requirements and compatibility

Unity	6000.3 (Unity 6.3 LTS) or newer. Tested with 6000.3.8f1; imports and compiles cleanly in 6000.6.0f1.
Render pipeline	Universal Render Pipeline 17 or newer. Tested with URP 17.3 and the Forward+ renderer. Built-in RP and HDRP are not supported.
Platforms	Platforms with compute shaders: Windows, macOS, Linux and consoles. Tested on Windows with Direct3D 12. Mobile, WebGL and XR are not supported in this version.
URP settings	Nothing to set. The water asks each camera for the depth and opaque textures it needs.
Dependencies	Burst and Mathematics, which URP already installs.
Input	The water reads no input. The demo scripts work with both the Input System and the legacy Input Manager.
Physics	Buoyancy uses the built-in 3D physics (Rigidbody and colliders).

3. Package contents

```
HavoilaStudio/DropInWater/  
  Runtime/      Water bodies, wave simulation, buoyancy, interaction, rendering  
  Editor/       Inspectors, Tools menu, shoreline baker, river handles  
  Shaders/      Ocean, river, underwater, mask and spray shaders, two compute shaders  
  Materials/    M_Ocean, M_Lake, M_River, M_Spray  
  Textures/     Foam and spray textures  
  Demo/         Demo scene (a pirate island), with its lighting and shoreline data  
    Terrain/    Its terrain and terrain layers  
    Textures/   Its textures (from CC0 materials; plants and sea life painted procedurally)  
    Meshes/     Its meshes (props, plants, sea life)  
    Prefabs/    Its prefabs  
    Materials/  Its materials  
    Shaders/    Its shaders  
    Scripts/    Demo scripts: camera, boat, autopilot, fish schools, ship sails, night fires, seagulls, crabs,  
                detail lighting, network clock  
  Documentation/ Markdown and PDF guide
```

The demo's own shaders, in `Demo/Shaders`, are made for the demo only: **Island Foliage** (wind, leaf flutter and light through the leaves), **Island Rock** (triplanar textures, so rocks need no UVs) and **Island Fish** (the fish swim; with the swim off it also draws the seagulls and the crabs).

The demo's lighting is baked, in Shadowmask mode: near the camera the sun casts realtime shadows, and further out the baked ones take over, with a quality level whose **Shadowmask Mode** is **Distance Shadowmask** (Unity's default). The light the sun and the sky bounce is baked into lightmaps (the terrain and the large buildings) and light probes (everything else, a probe per tree), and the jungle's canopy shades the ground under it. The grass, ferns and seagrass take the terrain's lightmap where they grow (**Terrain Detail Lightmap**, below). For the look of the screenshots, turn on **Bicubic Lightmap Sampling** (Project Settings > Graphics > URP > Lightmap Sampling Settings; off in a new project), which smooths the terrain's lightmap (one texel per metre), and set the URP asset's **Shadow Distance** to about 200 m: the grass and ferns are drawn to 170 m, and their sun shadows are the realtime ones. **Island Foliage** cuts the leaves out in a lighting bake through hidden copies of the material's **Albedo** and **Color**: after changing those on a foliage material, copy them in again before baking (see the shader's header).

4. Quick start

Open the demo

1. Open `Demo/DropInWater_Demo.unity` inside the package folder and press Play.
2. Hold the **right mouse button** and use **W A S D Q E** to fly (Shift = faster, mouse wheel = speed). Keys **1 - 8** jump to viewpoints: the island from the sky, the pirate cove, the beach, under water at the jetty, the river mouth, the jungle river, the hill lake and the open sea. **H** hides or shows the help.
3. Press **B** to take the helm of the rowboat in the cove (W S throttle, A D rudder). Press B again to let the autopilot row it in circles in the lagoon.

The demo is a pirate island shaped like a skull and crossbones. A coral reef rings it: the swell breaks on the reef, and the lagoons inside stay calm. The two eyes are lagoons. One is the pirates' cove, with a jetty, stilt huts, a ship careened on its beach and rowboats. The other is the river's mouth. A lake lies on the hill's shoulder, and its river runs down a cascade, then calmly through the jungle into that lagoon. A ruined watchtower and its cannons guard the pass into the cove, sea stacks line the skull's mouth, an old wreck lies on the reef, and a ship rides at anchor off the island. Things float in all three waters: on the sea, rowboats, barrels and crates in the cove, buoys along its pass and the ship at anchor; logs on the river and on the lake. Under water, schools of reef fish swim among corals, sea fans, sponges and seagrass, and an anchor, a treasure chest, amphoras and a cannon lie on the bottom by the jetty and the wreck.

Add water to your scene

1. **Tools > Havoila Studio > Drop-In Water > Create > Ocean**. The ocean sits at the height of its GameObject and reaches the horizon. It also adds a **Water Interaction** component, for ripples and wakes.
2. If the scene has a terrain, select the ocean and press **Bake Shoreline**, the large button right under its Material (or **Tools > Havoila Studio > Drop-In Water > Bake Shoreline of Selected**). The waves now shrink and break along the coast, and run up the beaches.
3. Shape the sea in the ocean's Inspector: **Water** sets its colour and reflections, **Waves** the sea state (wind and swell), **Shoreline** what the coast does to it (swash, foam, spray). Each part that can be turned off has its own switch, and **Quality** holds what it costs (section 6).
4. For a lake or a river: **Tools > Havoila Studio > Drop-In Water > Create > Lake / River**. The new object appears at the centre of the Scene view (a river over a terrain: just under its ground, where it digs its bed). Size the lake's box to reach under the shores. For a river, place its points from the source to the mouth.

The same items are in **GameObject > Havoila Studio > Drop-In Water**, and in the Hierarchy's right-click menu, which parents the new water to the selection.

Make things float

Select objects with colliders and choose **Tools > Havoila Studio > Drop-In Water > Selection > Make Selected Float**. It adds a Rigidbody and a **Floating Body** (and a Box Collider when there is none). The objects now float on whatever water they are in, with a mass computed from their volume and density.

5. How it works

Part	What happens	Where
Waves	Each Ocean and Lake builds a wave spectrum from its settings and evolves it with FFTs, one per cascade.	GPU compute, every frame
Gameplay copy	A low-resolution copy of the same waves is computed for the current time. Height, normal and velocity queries read it.	CPU, Burst jobs
Surface	A grid centred on each camera, with detail that decreases with distance, is displaced by the waves. It reaches the horizon.	GPU, one draw per body and camera
Shading	Absorption, scattering, refraction, reflections, foam and caustics, from the camera's depth and opaque textures.	Transparent queue (Transparent-100; rivers right after, Transparent-99)
Underwater	Rendered when a camera is at or below a surface: a mask of the surface, then a full-screen pass.	GPU, only when needed
Buoyancy	Sample points inside each body's colliders are tested against the water every physics step.	CPU, one Burst job per water body
Interaction	A square of water around the camera is simulated with an exact spectral solver. The result is added to every surface inside it.	GPU compute

- **No setup in URP.** The water hooks into URP's camera callbacks. It draws itself for every camera and requests the depth and opaque textures for the cameras that see it.
- **Several bodies of water.** A scene can have one ocean and any number of lakes and rivers. `WaterBody.Find(position)` returns the water at a position (rivers first, then lakes, then the ocean). Floating objects, cameras and the underwater effect all use it.
- **Deterministic.** The waves depend only on the settings, the seed and `WaterClock.Now`. The same time gives the same waves on every machine, at any frame rate.

6. Water bodies

Every water body's Inspector starts with the Havoila Studio header and a **Documentation** button. Hover any field for a tooltip.

The Ocean and Lake Inspector starts with the **Material**, then is split into sections: **Quality** (what the water costs), **Water** (its look, with its caustics, foam and whitecaps), **Waves** (the sea state), **Shoreline** (what the coast does to the water: the swash, the shore foam and the spray) and **Underwater**. Quality sets how much detail is spent on the water, the other sections what it looks like: a scene can be fitted to a platform without changing its look.

- **Bake Shoreline**, the large button right under the Material (tinted until the water is baked), bakes the ground around the water (section 8): every selected Ocean and Lake. The line under it says what is baked, or what the water lacks without it, and warns when a lake's box has grown past its bake.
- Each part that can be turned off has a switch in its header, down to each effect inside a section. A part switched off keeps its settings, greyed out, and so does everything it holds: with the **Shoreline** off, its swash, foam and spray are off too.
- **All ON / All OFF** set every switch at once. All off leaves a flat, still water with only its look, to judge it; then switch the parts back on one by one.
- **Expand all / Collapse all** open or close every section. The sections start collapsed and stay as you leave them for the rest of the editor session; a section greyed out can still be opened.
- A section's header shows a short status on the right: the size of the simulation, the height of the waves, whether the shoreline is baked, the droplets of spray alive, or why a part does nothing (not baked, no waves, the swash's foam instead).

Ocean

An infinite sea at the height of its GameObject; X, Z and rotation do not matter. Put one per scene.

Section	Settings
Material	The material asset (shared by every water using it; section 7).
Quality	Wave simulation: Resolution and Cascade Count of the GPU waves, Largest Tile Size (longer than the longest waves), Gameplay Resolution of the CPU copy used by queries and buoyancy, Loop Period (the waves repeat exactly after it). Surface mesh: Vertex Spacing near the camera, Grid Resolution , Levels (each doubles the distance covered), Morph Range . Reflections: Screen Space Reflections , SSR Steps , SSR Max Distance (on the material). Caustics: Caustics Resolution (the detail of the caustics' ripple net, simulated on the GPU: Off, 128, 256 or 512; Off leaves the broad light of the simulated waves only). Foam: Foam Detail (how many sizes of holes the foam draws up close: Low, Medium or High; lower costs less on slower GPUs, and far away it draws fewer by itself). Underwater: Light Shaft Samples , Beam Shadow Samples (Point & Spot Lights > Shadows in the Water). See section 13.
Water	The material's colour and clarity and surface (section 7), and Edge Smoothness (the water depth over which the edge of the water fades in where it meets the ground and objects: higher = softer); the Reflections of this body: Skybox , Intensity , Blur , Sun Highlight (section 7)
Water > Caustics (switch)	The moving net of sunlight the water focuses on the bottom and on objects under the water. It moves the sunlight only, not the sky's light: it fades with a low or weak sun, takes the sun's colour (warm at sunset), and less light crosses the surface with a low sun. Look : fills the settings below with a ready-made look (Lagoon, Pool, Sandy Shallows, Sharp Net, Soft Sand, Rainbow Bands; can be undone), to fine-tune afterwards. Intensity (1 = the light as the water focuses it), Cell Size (m: about how wide the cells of the net are; 0.1-0.2 the fine net of a pool, 0.3-0.8 a sandy lagoon, 1.5-4 broad soft bands), Focus Depth (m: where the net is sharpest; shallower water shows softer, wavier light, deeper water a net blurred by the sun), Sharpness (0 soft glowing patches, about 0.65 as sharp as the real sun allows, 1 razor-thin lines), Dispersion (rainbow edges along the lines: 0 white light, 1 real water, 5-10 the fringes seen in photos; wider with a low sun and deep water), Fade Depth (m). Under the More heading: Alignment (0 round cells, 1 long bands across the wind), Cell Variety (0 cells all alike, 1 large and small cells mixed), Speed (1 the real speed of ripples that size, 0 frozen; also follows Time Scale), Depth Consistency (0 as in reality, 1 the same net at every depth), Wave Lensing (how much the simulated waves gather the net into broad bands and shape the light shafts), Follow Wind (on: a stronger wind gives a sharper net nearer the surface, and with the Waves off there is no net). Far away the net fades out by itself before it would sparkle.
Water > Point & Spot Lights (switch)	Point and spot lights on the water, over and under the surface, with their shadows and cookies: Highlights (on the surface, reflected or coming through it) and Scattering (the light they spread in the water and on its foam, and the glow and beams around them under water); 1 = physically based. Shadows in the Water (off by default): objects cast shadows in that light and cut shadowed gaps in the beams. Section 7.
Waves (switch)	Wind: Speed (m/s at 10 m: 3 light air, 8 moderate breeze, 14 near gale, 25 storm), Direction (where it comes from, degrees clockwise from world +Z), Fetch (km of open water the wind blows over: short fetch = young, short, steep waves), Alignment (0 confused sea, 1 long-crested). Swell: Height (significant height, 0 = none), Period , Direction (where it comes from, degrees clockwise from world +Z), Spread . Shape: Choppiness (sharp crests), Water Depth used for the wave shape; under the More heading: Peak Sharpness , Short Wave Cutoff , Time Scale , Seed . Off: a flat, still surface (the ripples of Water Interaction only), without surf, swash or spray; once the whitecaps' trail has faded, nothing is simulated any more.
Waves > Whitecaps (switch)	Foam on the crests breaking in open water: Coverage (0.5 is calibrated on real seas), Lifetime , Gusts (how patchy: 0 as much foam everywhere, 1 patches of more and less foam, 2 very patchy; the patches also keep the waves' repeating tiles from showing as rows of the same whitecaps) and Gust Size (m). Their look is the foam's (Shoreline > Foam): fresh and thick on the crests breaking now, a thinner lace in their trails.
Shoreline (switch)	The ground around the coast, baked from the terrain (section 8): Baked Data and the Bake Shoreline button, Wave Depth , Breaking Depth . Off: the water behaves as if there were no shore, as without baked data (and without its foam and spray).
Shoreline > Swash (switch)	The waves running up the beaches, with their bores, sheet and wet sand (section 8); their foam is in Shoreline > Foam . Inside it, Spray Particles (switch): droplets thrown up by its bores (section 8).

Section	Settings
Shoreline > Foam (switch)	The foam along the shore and how every foam of the body looks. Its switch is the shore's foam's, the only one: off, none along the shore (the line of foam against the ground and objects, the surf, the white water and lace of the bores and the swash, the mark the swash leaves on the sand) and none of its cost, and the spray, which comes out of the foam, stops too; the whitecaps stay (Waves > Whitecaps, their own switch) and so do the wakes. How much foam there is stays with each source (Whitecaps; below, the line along the shore and the Surf, or the swash and its bores). Foam is a thin layer of bubbles with holes of clear water: as there is less of it, its holes grow and merge, the filaments between them break and the last clumps go; it never fades. Fresh foam (a crest breaking now, a bore's white water) is thick, bright and finely holed, with milky air in the water under it; old foam is a thin lace the water and the sand show through. Look: fills the settings below with a ready-made look (Natural Sea, the default; Tropical Surf, Gale, Calm Lake, River Rapids, Organic Foam, and Stylized, an opaque white polygon lace with crisp edges; can be undone), to fine-tune afterwards; it also sets Swash Hole Size (River Rapids leaves it). Pattern: where the holes come from: Procedural (the default: computed, the exact amount of foam, clean at every distance) or Texture (the red channel of a texture you can replace: Pattern Texture, and Holes Across, how many of its largest holes it holds across; with a texture not yet equalised, Make Equalised Copy; see below). Hole Size (m: about how wide and how far apart the largest holes are; 0.3-1 on the open sea, 0.1-0.3 in rapids), Swash Hole Size (m: the same for the foam of the bores and the swash, usually smaller: 0.1-0.4; greyed out without the Swash), Edge Softness (0 crisp cut-out edges, 0.25-0.5 real foam, 1 soft clouds), Spread (how foam thins out as there is less of it, as it ages: 0 it stays in compact opaque clumps that shrink, 1 it spreads into a wide, thin, see-through lace before it goes), Brightness (how much light thick foam reflects: 0.55-0.65 measured sea foam, 0.9-1 snow-white), Thickness (bubble layers of the densest, freshest foam: 1-3 a film of bubbles, 10-15 surf foam, 25 and more opaque breaking crests; thinner foam lets the water show through and is dimmer), Color (tint of the foam before light, also the swash's: near white for sea foam, creamy for foam held by organic matter). Under the More heading: Scale Variety (0 holes of one size, 1 holes from Hole Size down to a seventh of it: big ones in old foam, fine bubbly ones in fresh foam), Hole Variety (0 holes of one size all alike, 1 from about a third to three times their mean size), Cell Shape (0 round holes that grow and merge, 0.2-0.4 a connected lace of filaments, 1 polygon cells with walls of even width), Patchiness (patches of denser and thinner foam a few holes across), Stretch (how much longer than wide the holes are along the wind, more in stronger winds, round in a calm), Wind Streaks (old foam gathered into long rows along the wind, from about 5 m/s and well marked from 14 m/s; their spacing follows the wind), Relief (how much thick, fresh foam stands up from the water, lit on one side and shaded on the other), Translucency (sunlight shining through thin foam when looking towards the sun), Bubbles and Bubble Size (cm: the bubbles seen within a few metres, the bright walls between them and the sun glinting on fresh ones), Underwater Bubbles (the milky glow of the air the breaking water drives under the surface, tinted by the water itself: turquoise in clear water). With Pattern = Texture, Cell Shape and Hole Variety are hidden: they shape the procedural holes only. Far away each size of holes fades into its average by itself before it would sparkle, so the foam is as bright at every distance. Under the Along the shore heading: Width (depth of water, m) and Intensity of the foam along shores and around objects; Surf, the foam of the waves breaking in shallow water. With the Swash on, its own foam replaces those three: they are greyed out, and the settings under the Swash and bores heading apply instead (the white water of the bores, the front line, the lace, the backwash streaks, the swash mark: section 8; greyed out without the Swash).
Underwater (switch)	Waterline, Light Shafts and their Light Shaft Intensity

The foam's pattern. With **Pattern** = Procedural (the default), the holes of the foam are computed: the foam covers exactly its amount, and far away each size of holes fades into its average. With **Texture**, the holes come from **Pattern Texture** instead; the rest is the same: the amounts, the freshness, the thickness and the light, the looks, the distance. Without a texture, the procedural pattern is drawn (the Inspector says so); switching to Texture with the field empty fills in the supplied **T_FoamHoles** (**Textures** , 8 holes across).

- **What the texture holds.** Its red channel is how long each point of the foam lasts: where it is high the foam lasts longest, where it is low the holes open first. It tiles seamlessly, and its values are spread evenly over 0-1 (equalised), so that the foam covers exactly its amount. Import it with sRGB off, mipmaps on and no compression (Single Channel, Red). **T_FoamHoles** is all of this.
- **Holes Across** is how many of its largest holes the texture holds across, 1 to 64 (a power of two, so the pattern stays seamless): **Hole Size** sets how big each is on the water. Each size of holes reads it once, turned and shifted so that the sizes do not line up.

- **Make Equalised Copy** (shown with a texture not yet equalised: neither T_FoamHoles nor a copy it made) writes `<name>_Equalised.png` next to the texture: its red channel (where the red is the same everywhere, its brightness, else its alpha) spread evenly over 0-1, in the same order, so where it was higher the foam still lasts longer; flat areas are ordered from their darker side to their brighter one. It imports the copy with the right settings and uses it. It reads PNG, JPG and EXR files up to 2048 x 2048: save other formats as PNG first.
- Cell Shape, Hole Variety and the bubbly edges belong to the procedural pattern; the bubbles seen up close stay procedural, and so do the spray's droplets: they leave from the procedural pattern of the same amount of foam (section 8).

Lake

Wave water inside a box, at any height: lakes, ponds, pools, flooded caves. It has the Ocean's settings, plus a **Lake** section under the Material:

Setting	Use
Size	Width and length of the box (m), centred on the object and rotated with its Y rotation. Make it reach under the shores: the terrain hides the rest. With Shape = Basin it only bounds the lake.
Shape	Basin (default): the water fills the lake's basin, the ground lower than the water that the lake reaches from its centre (this object), inside the box. Lower ground beyond its rim (a valley below a crater lake) stays dry, though the box reaches over it: it is cut along the middle of the rim, where the terrain hides the cut. It needs the baked shoreline (section 8; without it, the whole box) and follows the water level without baking again. A river flowing out of the lake down a valley lower than the lake is dammed across the valley where it has taken over from the lake (past half its Source Blend Length : keep that short where the river drops away from the lake, or the lake spills down the valley before the dam). The section's status gives the lake's area, and warns when it reaches the edge of the box. Box : the whole box, wherever the ground is lower than the water.
Depth	How far below the surface the lake counts, for cameras and floating objects.
Edge Fade	Distance over which the waves fade out before the edges of the box.
Fetch From Size	On by default: the wind blows over the length of the lake, so a pond ripples and a 5 km lake gets real waves (the wind's Fetch is then greyed out).

The default material, M_Lake, is fresh water: greener, with shorter visibility than the sea. A new lake's foam has the **Calm Lake** look: thinner, with smaller holes and little air under it.

River

A flowing ribbon of water along a spline, from the source to the mouth. Its Inspector is laid out like the Ocean's and Lake's: the material, the **Bake River Bed** button with its status, then the sections **Quality** (what it costs: its ripples' simulation, its mesh, the reflections, the caustics' and the foam's detail, the light shafts; as on the Ocean and Lake), **Water** (its look: the material's settings and reflections, section 7; in it **Flow**, **Ripples**, **Foam**, **Caustics** and **Point & Spot Lights**), **Path** (its points; in it **Mouth** and **Source**, **Ground** and **Bed**) and **Underwater**.

- **Scene view.** Move the points with the handles, drag the blue dots to set the width, and **Shift + click** on the ground to add a point at the end (with Carve Bed on, just under the ground as it was before the river dug it). With Carve Bed on, a brown cone hangs under the middle of each segment, at the bottom of its bed: drag it down to deepen the bed there, up to raise it (**Ctrl**: by 0.1 m; double-click: back to the river's Depth). While you hover or drag it, a cross-section shows the bed it asks for (brown), the terrain as it is (grey) and the water (blue).
- **Fit Width To Terrain** (Path section, or **Tools > Havoila Studio > Drop-In Water > Fit Selected River to Terrain**) widens or narrows each point until both edges are under the banks: the ground the bake takes (the terrains as they were before the rivers dug their beds, and the colliders on Ground's layers). Where the river meets a lake or the sea (a point over their water at their level, or their water beside it), there is no bank to find: those points keep their width, set by hand. Their width dots are orange in the Scene view. **Reverse Flow** swaps the source and the mouth.

- **Flow.** The water flows along the spline. Its speed is $\text{Base Speed} + \text{Slope Speed} \times \sqrt{\text{slope}}$, up to Max Speed, for the river as a whole; across each section it is faster where the water is deeper (as in open channels: $\text{speed} \sim \text{depth}^{2/3}$), its mean through the section being the river's speed there, and faster through narrower or shallower sections. **Bank Speed** is the slowest it gets at the water's edge. Ripples and foam are carried by the flow, and the water turns white where it is fast, more where it is fast and shallow, and behind rocks (material: Rapids Speed, Rapids Foam). All this comes from the river's flow map, built with its path: see Bake River Bed below.
- **The pattern carried by the flow.** The ripples and foam are carried in two copies, each for half a **Flow Cycle** on average (material, Flow section). Where the water's speed changes quickly across or along the river (faster in the middle than at the banks, around a rock), a copy carried that long would stretch into streaks and arcs: there the pattern follows the closest flow that changes no faster than the Flow Cycle allows, so it is never sheared more than half its size; elsewhere it follows the water exactly. The Flow section says over how much of the river it does. A longer Flow Cycle carries the pattern further between resets but evens it out more; 2 s is a good start. The pattern is carried along the river's own lines, so bends and changes of width do not stretch it either.
- **Foam** (switch). How the river's foam looks (its white water, the wakes, the foam along its banks) is on the component, set as on the Ocean and Lake (**Shoreline > Foam**, above; off: no foam on the river at all, and none of its cost): **Look** (a new river has **River Rapids**: small holes stretched along the flow, strongly aerated water under the churn), **Pattern** (with Texture: **Pattern Texture**, **Holes Across**, **Make Equalised Copy**), the same settings but Wind Streaks (**Stretch** is along the flow), and **Foam Detail**. Under **Along the banks (material)**: **Shore Foam Width** and **Shore Foam Intensity**, shared by the rivers using the material. How much white water there is stays in Flow (material: **Rapids Speed**, **Rapids Foam**): the faster the water, the fresher its foam, thick and finely holed at drops and hydraulic jumps.
- **Never Uphill.** A point higher than the one before it is lowered, so the water never flows uphill.
- **Shape. Segment Length** and **Cross Segments** of the mesh (in **Quality**).
- **Riverbed.** With **Carve Bed** on (a new river has it on; a river from an older scene has it off), the river digs its bed into the terrains under it and keeps it in step as you edit the river: points, widths, Depth, the segments' offsets and the bed's shape. **Depth** is how deep the bed is under the surface at its centre, for the whole river (2.5 m for a new river); each segment adds its own **Bed Offset** (in the Points list, or with the cones in the Scene view), exact under the middle of the segment and eased from one to the next. At a free end the bed rounds off up to the water; where the river fades into a lake or the sea it fades out with it, cutting a sand bar down to their level, never a trench into their floor. **Bed Shape: Flatness** (0: rounded, deepest in the middle; 1: a flat bottom with steep sides), **Bank Slope** and **Bank Height** (the banks rise from the water's edge at that slope up to that height, then steeply, at 70 degrees, up to 30 m above the water). The water's edge is a little inside the ribbon (1.5 m on a wide river), so the ribbon's edges stay under the banks.
 - The bed follows 0.2 s after your last change, and at once on undo and redo. While you drag a handle it follows live if carving takes under 20 ms, else when you release the mouse (about 0.1–0.15 s for a 1 km river on a 4097 x 4097 terrain).
 - **Your own edits are kept.** Sculpt the terrain as usual: in a bed, your edits (a bar, a pool) stay relative to the bed, and follow it when it gets deeper; beside it, they are the ground. Turning Carve Bed off, or deleting the river, gives the ground back exactly as it was, with your edits; undoing the delete digs the bed again. Disabling the river, or closing its scene, keeps its bed.
 - What the bed needs to stay in step is kept next to each TerrainData, in "**River Beds.asset**" (the ground without the rivers, your edits, and each river's cut): commit it together with the TerrainData. A terrain found back at its ground without the rivers (an older version of it) is dug again.
 - The Inspector says where the bed is carved, how deep, your edits kept, and warns where the water stands above the ground beside it: carving only lowers the ground, so lower those points (or raise the ground there). **Recarve** digs the whole bed again; **Forget Edits** drops your edits of this river's bed; **Detach Bed** keeps the bed as it is now as part of the ground and turns Carve Bed off; **Select Store** shows the asset. Forget Edits and Detach Bed cannot be undone.
 - A carved terrain that contributes to baked lighting needs its lighting generated again. Trees and grass in the bed are not moved.
- **Bake River Bed** (the button at the top, **Tools > Havoila Studio > Drop-In Water > Bake River Bed of Selected**, or **Bake All Water in Scene**) samples the ground under the river on its flow grid (**Ground > Texel Size**, 0.5 m by default, across and along): the terrains (**Terrains**) and the colliders on **Collider Layers** (Default for a new river): rocks, piers, props standing in the water, a scanned landscape's mesh, but not floating or moving objects (a non-kinematic RigidBody). An object standing in the water on another layer is named under the button, with **Add To Collider Layers**. Colliders more than **Clearance** above the water (bridges, roofs, overhangs) are not the river's ground. With it, the river's depth, how

fast it flows across it and where it turns white follow the real bed: rocks standing out of the water are dry, the water goes faster beside them and foams behind them, and with **Steer Around Rocks** on the water turns around them (a 2D solve of the flow through the bed, kept in the bake) and goes on past them, leaving slow, dead water in their lee instead of closing in behind them. Queries return that velocity, so floating objects go around rocks too, and the river does not hold a camera or an object below its baked bed or inside a rock.

- **Auto** (beside the button, on for a new river) bakes on the fly: as you edit the river, the rows it has moved away from are sampled again (at once on a small river with terrains only, else when you release the mouse); as you sculpt the terrain, or a river digs its bed, the rows over the change follow; and as you add, move or remove an object on Collider Layers, the rows where it stood and stands follow too (when you release the mouse). Without a bake, or where the river has moved since and Auto is off, the river assumes its bed: the one it digs (Carve Bed), else a rounded channel of its Depth.
- The bake is saved next to the scene, like the shorelines (`<Scene>/River_<name>.asset`); a river in a scene never saved keeps it in memory until you save the scene. A copy of a river makes its own when baked. A bake made with the button can be undone.
- **Scanned ground** photographs the water's surface, not its bed. **Scan Tolerance** keeps ground up to that height above the surface as water, and **Shore Depth Growth** makes the water at least that $x \sqrt{\text{distance to the bank}}$ deep (try 0.3 and 0.5 for a scan).
- The status under the button says what is baked, how deep and fast the river is, its rocks, what changed since the bake (the rows the river moved from, the terrain edited), and where the water reaches the edge of the ribbon (widen those points). **Preview** unrolls the bake along the river, source on the left: depth (orange where not from the bake), speed, direction and white water; hover it to read them.
- **Show Flow in Scene** (Ground section): over the selected river, four lines along its water, coloured by its speed (blue slow, red fast), with a small arrow about every 4 m. The Preview shows the rest: depth, rocks, direction across, white water.
- The water fades out at its edge where the depth is known (baked, or the bed it digs), so the shoreline is smooth even where the terrain's grid draws steps.
- **Mouth and Source.** Where the river flows into a lake or the sea, or out of one, it blends into their water (**Blend With Water**, on by default). Over its last **Mouth Blend Length** metres (its first **Source Blend Length** metres where it leaves a lake or the sea; 24 by default) the river first joins their water level, then fades out into their water, from its sides first, like a tongue of river water spreading out (at the mouth, from wherever it already lies at their level: a river whose last points lie under the water starts fading sooner, and never dips under their level): its colour and ripples give way to theirs, and their waves, calmed under its current, come back. Under water it blends the same way: swimming from the river into the lake or sea, the river's colour, clarity and light turn into theirs as its water gives way to theirs, without a jump. Nothing to set: the river finds the Ocean or Lake its first and last points lie in, when they lie over their water from 1.5 m under their level to 1 m above it. Where the river has not started to fade, the lake or sea is not drawn there, nor further up the river wherever their water would show inside it (a channel dug below sea level, from their shoreline data; without it, down to the river's **Depth**), so a lake's box may reach over the river where it leaves the lake.
- **Surface. Ripples** are small wind waves carried by the flow, with their own small simulation (its resolution and tiles in **Quality**). Their **Wind Direction** is relative to the river: 0 (the default) blows them downstream, with the current; 180 sends them upstream, against it (where the current is slow, they then seem to make the water flow backwards); ± 90 across. Underwater and Caustics work as on the ocean; the caustics' net is carried by the flow too (its **Caustics Resolution** is in **Quality**, 128 by default). Under a river (the camera in it) its caustics are carried by the flow beside the camera, the whole view at that speed; where it blends into a lake or the sea, theirs take over as their water does.
- Queries return the flow velocity, so floating objects drift downstream on their own. Outside the river there is no water: `SampleHeight` returns -1000000.

To connect a river to a lake or the sea, start or end it a little inside their water, at their level: its first Source Blend Length or last Mouth Blend Length metres fade into them. At the mouth, end the river about half a Mouth Blend Length past the point where it reaches their level, or lengthen the Mouth Blend Length to reach back there (a river spreading out over a beach or into a lagoon); at the source, keep the Source Blend Length short where the river drops away (a cascade). The demo's river blends this way into the sea, and out of the lake.

7. Appearance: the water material

The Ocean and Lake use the **Havoila Studio/Drop-In Water/Wave Water** shader; rivers use **River**. Edit the material in the **Water** section of the Ocean and Lake Inspector (its reflection quality in **Quality**), in the River's **Appearance** foldout, or on the material asset. Material settings are shared by every body using that material; **Reflection Intensity**, **Reflection Blur** and **Sun Reflection Intensity** are per component (on the Ocean and Lake: **Water > Reflections > Intensity**, **Blur** and **Sun Highlight**). The foam's look is on the component for every body (the Ocean's and Lake's **Shoreline > Foam**, the River's **Foam**; section 6). The Ocean and Lake have their shore settings on the component too: **Water > Edge Smoothness** and **Shoreline > Foam**; the material's are for rivers.

Group	Settings
Water Body	Scattering Color (light coming back from inside the water), Absorption Color (colour left after travelling through the water), Absorption Distance (visibility, m), Crest Scattering (light through thin wave crests)
Surface	Extra Roughness , Refraction Strength , Refraction Smoothing (higher = steadier refraction that follows only the larger waves), Rain Ripples (the rain's rings, with a sky that publishes its rain; 0 = none), Shore Fade (soft edge where the water meets the ground; rivers only)
Reflections	Screen Space Reflections on/off, SSR Steps , SSR Max Distance . Beyond them, reflection probes and the sky are reflected.
Foam (rivers, along the banks; the Ocean and Lake use Shoreline > Foam)	Shore Foam Width , Shore Foam Intensity . How the foam looks is the component's (River > Foam).
Flow (River)	Flow Cycle , Ripple Strength , Rapids Speed , Rapids Foam

- **Skybox** (component, on by default; the River's **Skybox Reflection**): the water reflects the skybox as it is now, and takes its ambient light from it. It captures the sky itself, again whenever the sky material, any of its settings or the sun changes (at most ten times a second while it keeps changing, as in a day-night cycle), so it follows the sky without generating the lighting. The ambient light (on the water, its foam, its spray and under it) is the light the captured sky gives, times the Lighting window's ambient **Intensity Multiplier**, when the scene's **Environment Lighting Source** is Skybox; with Gradient or Color, the water keeps those. Off: the water reflects the scene's reflection probes instead, as other objects do (the environment reflection generated in the Lighting window, made with the sky and sun of that moment, or your local probes), and takes the scene's ambient light. Either way, what is on screen is reflected by the screen-space reflections on top. A sky that draws its own environment every frame can hand it to the water instead (Drop-In Sky & Weather, sold separately, does with its Environment Lighting on): the water then reflects that sky as it is, its clouds as they move, without the sun's and the moon's disks (their lights make the highlights), and captures nothing itself.
- **Fog**. With a sky that publishes its fog (Drop-In Sky & Weather, sold separately), the water is fogged by it like the rest of the scene, mist and shafts of light included, and so are its spray and what it reflects; otherwise it uses Unity's fog (Lighting window, Environment > Fog). Nothing to set.
- **Rain**. With a sky that publishes its rain (Drop-In Sky & Weather, sold separately), rings spread where the drops fall on the water. They are drawn on the surface itself, so they ride the waves back and forth as the foam does, and drift downstream with a river's flow. Falling snow rings the water too, more softly. They show near the camera, as far as the sky's Splash Distance. Farther away, where they are too small to see, they roughen the surface: duller reflections and a broader glint of the sun. The material's **Rain Ripples** sets how strong they are (1 = natural). Without such a sky, nothing changes. The water also publishes where its surface is around the camera (`_HavoilaWaterHeights`: its height seen from above, at rest), so such a sky's rain and snow stop at the surface instead of falling through to the bottom, and splash on it.
- **Point & Spot Lights** (component, on by default; the Ocean's and Lake's **Water > Point & Spot Lights**, the River's **Appearance > Point & Spot Lights**): URP's point and spot lights light the water like the sun does, with their shadows and cookies: their highlights on the waves, their light coming through the surface (a lamp in the water seen from above as a refracted point of light, a lamp above the water seen from under it through the surface, with total internal reflection beyond the critical angle), the foam they light, and the light they spread in the water under the surface, from a lamp

above it or in it, absorbed along the way. On beaches the swash takes them too: its sheet and bores like the sea, its foam, and the film of water it leaves on the sand, which reflects a lamp (and the sun) as sharply as it is smooth (a bright streak on the sand just after the backwash). Under water, the water glows around them, fading with the absorption, and a spot light shows as a beam, with its cone and its cookie's pattern. **Highlights** and **Scattering** scale these (1 = physically based). The glow seen under water needs the Forward+ renderer (the URP default; Deferred+ too); the rest works with every renderer, within URP's per-object light limit with Forward. Off: the sun only, cheaper with many lights.

- The light's shadows always apply to its highlights and to the foam. **Shadows in the Water** (off by default) applies them in the water too: objects between a light and the water shadow the light it spreads there, and cut shadowed gaps in its glow and beam seen under water, such as a lamp shining through a grate or a hull in front of a diver's spot light. Each shadowed light's beam is then sampled at **Beam Shadow Samples** points (Quality, 4 - 32, 12 by default) with a shadow lookup each, spaced by its light and jittered every frame like the sun shafts: more samples give sharper, steadier shadow edges in the beams, at a higher cost. It only concerns lights that cast shadows: set the light's **Shadow Type**, and enable **Lighting > Additional Lights > Cast Shadows** in the URP asset.
- A spot light's beam is cut exactly to its cone before it is sampled, so its edges stay clean whatever the number of samples.
- **Reflection Intensity** (component, 0 - 2) scales the reflections of the sky and the scene (reflection probes and screen-space reflections) on that body only, on the wet sand of its swash and on its foam (the foam reflects the sky like the water, blurred: lit by neither the sun nor the ambient light, it still shows what the water around it reflects, instead of turning black). 1 is physically based: weak looking down, strong at grazing angles. 0 removes them, so only what is under the water shows; above 1 they get stronger. The sun highlight is not affected.
- **Reflection Blur** (component, 0 - 1) blurs the reflections of the sky and the scene on that body, on top of what its ripples blur, as on a rougher surface. 0 is physically based; towards 1 the reflections turn into a smooth blur (the screen-space reflections hand over to the reflection probe). The sun highlight is not affected. On the Ocean and Lake, the swash and the wet sand have their own: **Shoreline > Swash > Reflection Blur**.
- **Sun Reflection Intensity** (component, 0 - 10) scales the direct sun highlight of that body only: on the water, and on the Ocean and Lake on the film of water the swash leaves on the sand, where the sun's shadows cut it. 1 keeps the physically based glint; 0 removes it. The sun light, the sky reflection and the material are unchanged. On the wet sand the glint, like the reflections, is as sharp as the film is smooth: a mirror on the water still standing after the backwash, spreading out and dimming where the film thins out into damp sand.
- **The sun's cookie.** The water follows the cookie of the scene's main light, as the objects around it do: cloud shadows a sky asset casts through the sun's cookie, or a cookie texture of your own, darken its glints, the light scattered in the water, its foam, the glint on the wet sand, the caustics and, under water, the light shafts and the water's glow, and its spray (which also takes the sun's shadows). Nothing to set; without a cookie, nothing changes.
- Colours are picked in sRGB like every colour field. The shader works in linear space.
- To give two bodies different looks, duplicate the material and assign the copy to one of them.

8. Shorelines

Select an Ocean or a Lake and press **Bake Shoreline** at the top of its Inspector (or in its **Shoreline** section), or choose **Tools > Havoila Studio > Drop-In Water > Bake Shoreline of Selected**, or **Bake All Shorelines in Scene**. The component's context menu has **Bake Shoreline** as well. The ground heights of the terrains around the water are stored in an asset next to the scene (`<Scene>/Shoreline_<name>.asset`). With them:

- waves shrink in shallow water and stay off dry land, on the GPU and in the gameplay queries alike (**Wave Depth**: depth, in wave heights, where they keep 63 %);
- waves break into surf where the water gets shallow (**Breaking Depth**, and **Shoreline > Foam > Surf**);
- foam lines the shore (**Shoreline > Foam**), and the water fades softly against the ground (**Water > Edge Smoothness**, with or without baked data; with the swash, the sea fades into the wet sand or the thin sheet, over at least half the depth the swash starts at, and follows the backwash down, so the two meet without a line at the still water line);
- the ocean no longer counts as water on dry land below sea level: no underwater view in a valley, no buoyancy on a beach;
- a lake keeps to its basin (Lake > **Shape**): no water over lower ground beyond its rim.

Beyond the baked area the ground is unknown: past an edge that is under water, the sea floor is taken as going on down (1 m every 10 m), so the open ocean keeps its full waves; past an edge on land, it stays land, so an ocean beyond it stays calm, with a shoreline, out to the horizon. So **Bake Shoreline** bakes an ocean a little past its terrains, where the ground is unknown too: past a terrain's border under water, the sea floor goes on down the same way; past dry land, the sea is deep (the land ends with the terrain). The edges of the data are then always water. With an area of your own (from a script, below), keep the coast and its shallows inside and every edge under water: the baker warns in the Console when an ocean's area ends on dry ground, and says where. **Wave Depth** therefore only matters where the baked water is shallow compared with the waves: if it changes the waves everywhere, the sea floor of the baked area is shallow everywhere.

The **Shoreline** switch turns all of this off at once, without losing the baked data: the water then behaves as if there were no shore.

The bake follows the terrain while you sculpt it in the Editor: after each stroke, or a script's `TerrainData.SetHeights`, the part of the bake under the changed terrain is baked again (its area and resolution are kept), for every water in the open scenes; undoing the edit brings it back the same way. Out of date, a bake keeps the old ground: on a raised beach the swash runs up inside the sand and throws its spray out of it. Changes made while the water's scene was not open (by another scene, or version control), and terrains added, moved or resized, are found when the scene opens or the water is selected (large changes: the bake is compared on a grid of about 65,000 points): the Console warns, the Shoreline section says "terrain changed", and the Inspector offers **Update From Terrain**, which bakes that part again. Bake again after adding, moving or resizing terrains. Where a terrain the bake was made from is not loaded (a tile in another scene), the bake keeps what it took from it. A bake that took the static colliders only follows the terrains: bake it again after moving them.

From a script (Editor only), `ShorelineBaker.Bake(water, area, texelSize, includeColliders)` bakes, `ShorelineBaker.Rebake(water, area)` bakes again part of an existing bake, and `ShorelineBaker.FindStale(data)` says where the terrains no longer match a bake.

Swash: waves running up the beach

With baked data, each wave is followed from the breaker line to the top of the beach as one body of water. It breaks where the water gets as shallow as **Breaking Depth** and rolls in as a bore: a real raised front of white water (its amount is **Bore Clumps** times **Intensity**, in **Shoreline > Foam** under **Swash and bores**), moving at the speed of waves in shallow water and leaving foam behind. It reaches the still water line exactly when its run-up starts, and a strong bore makes a long run-up. The front climbs the beach, slowing down, its bore collapsing into a thin sheet that refracts the ground under it. Then the water drains back faster, pulling the foam left on the sand into streaks down the slope and drawing the sea down below the still water line until the next wave floods it back, and the sand it leaves stays dark and shiny for a while. Timing, strength and the fingers of the front vary along the shore. The water only runs where it can get to: dry ground behind a dune or a ridge (a hollow inland of a beach) is reached only by a run-up higher than the lowest crest between it and the open sea (a lake's water), so no swash, wet sand or spray appears there below that; ground below the water level behind a crest (a pool) keeps its water as before. This follows the water level (a lake raised past a crest floods what lies behind it), and **SampleDepth** reads it too: behind a crest it gives, negated, how high the water must rise to get there. With the swash on, its foam replaces the Shoreline's (**Shoreline > Foam** is greyed out): the bores replace the surf foam drawn on the crests of the waves, and the swash's water and foam replace the line of foam along the shore, including around objects standing in the water.

The **Swash** (in the Shoreline section) has a switch in its header, on by default; the effect needs the baked data and the waves. Its parts (**Sheet**, **Bore**, **Wet Sand**, under **Parts**) follow, then **Spray Particles**; the Bore and the Wet Sand have their own switch too, so each can be judged on its own. Its foam is with the rest of the shore's, in **Shoreline > Foam** (under **Swash and bores**; **Swash Hole Size** next to the foam's Hole Size): **Shoreline > Foam**'s switch is the shore's foam's, the swash's included.

Waves	Use
Run Up Height	How high the waves run up above the still water level (m). About 0.6 × the significant wave height is typical of sandy and pebble beaches. It alone sets how far the swash goes: it starts a little below the still water line (0.45 × the run-up) and climbs to the run-up. The height of the waves (wind, swell) does not change it; the bores and the surf follow the waves. (Scenes saved with the former Run Up , in significant wave heights, are converted when first drawn, to the same height for the waves they have then.)
Period	Time between two waves running up (s). 0 = the peak period of the waves.
Uprush Share	Part of each wave's cycle spent running up; the rest is the backwash.
Reflection Blur	Blur of the reflections on the swash and on the wet sand, independent of the sea's (section 7).
Variation > Height	How much the waves differ: each runs up (and breaks) up to this much higher or lower (0.35 = ±35 %).
Wave direction > Follow Wave Direction, Lee Side	With the switch on (default), the swash follows the direction of the waves; Lee Side is how much of it is left on shores facing away from them, such as the far side of an island (0 = none, 1 = the same all around). Off: the same swash on every shore, at times that vary along it (Variation > Timing). The waves come from the swell's Direction (the wind's when the swell is less than half as high as the wind sea): shores facing them get the whole run-up, bores, foam and wet sand, less and less round the sides, down to the Lee Side behind. How far round they still reach follows their spread of directions (Swell Spread , or the wind's Alignment). They also reach each stretch of shore when their crests get there, so on an oblique beach the waves run up in sequence along it, and they reach the far side of an island last.
Variation > Timing	How much the time the waves reach the beach varies along the shore, in wave periods. 0 = the whole shore at once.
Variation > Scale	Length of shore over which the timing changes (m); the height changes over half of it.

Sheet	Use
Front Depth, Deepening, Max Depth	Depth of the water behind its front (cm), how much deeper per metre behind it (cm/m), and at most (cm).
Backwash Thinning	How much thinner the sheet gets while it drains.
Swash Smoothness (min, max)	Softness of the front of the swash: width of its thin tip (m), where the water thins to nothing and the sand shows through. The front uses the lowest value while it moves fast (just after the bore reaches the beach, and at the end of the backwash) and the highest where it slows down and turns at the top of the beach, so it never switches from sharp to soft. Varies along the shore. 0, 0 = a sharp front.
Infiltration, Infiltration Edge	In the backwash the water also soaks into the sand: the sheet turns transparent as it comes down, completely when its front reaches the still water line, leaving shiny wet sand. Its thin upper edge goes first, over a width that grows as it drains: the Swash Smoothness plus up to the Infiltration Edge (m). 0 = the water only drains back into the sea.
Slope Follow	How much the surface follows the slope of the sand. At grazing angles every degree of tilt lifts the reflection by two degrees: at 1 the sheet reflects the sky above what the sea next to it reflects.
Ripples, Ripple Size, Ripple Speed	Small waves on the sheet, carried up and down the beach by its water as its foam is (its backwash streaks stretch them too): how steep they are (0 = a smooth sheet that mirrors the scene), the wavelength of the largest (m; smaller ones, down to half as long, come with them), and how fast they run across the water on top of being carried by it (1 = as fast as real ripples of their size in a few centimetres of water, 0 = they only ride along). They are roughest as the wave comes up, smooth in the thin tip of the sheet, and far away they blur its reflections instead, before they would sparkle.
Backwash Ripples	Ripples left by the end of the backwash, as a share of those of the uprush: the draining water calms down. 0 = it turns glassy.
Edge Fade	Depth over which the sheet turns clear at its very edge (mm), where it carries the water on up the sand. Its surface still reflects like the rest of the water, however thin.
Fingers, Finger Length, Finger Size	Tongues of thin water running ahead of the front: on/off, how far (m), how wide (m).

Bore (switch: off = the waves break as ordinary surf, with foam on their crests)	Use
Height	Height of the bore where it turns into the swash: 0 = none, 1 = $0.02 \times$ the significant wave height, 3 = the largest ($0.06 \times$ it). Taller further out; a bore never gets taller than about $0.8 \times$ the depth of the water (it would break). It only changes how tall the bore is, not how far the waves go. The bore is a smooth wave: its face spans a few cells of the surface mesh (Vertex Spacing), so the mesh and its shading draw it the same at every distance, without steps. (Scenes saved with the former Bore Height , in wave heights, are converted when loaded, to the same bore up to the largest.)
Duration	How long the water stays raised behind the front, as a fraction of the period.
White Water / Lace	How long the white water, then the lace it leaves, last (s). White Water 0 = none: the bore brings only lace.

Foam of the swash and its bores (Shoreline > Foam, under Swash and bores)	Use
Intensity	Overall amount of swash foam. Less foam opens its holes wider; it does not fade.
Swash Hole Size (with the foam's look, next to its Hole Size)	Size of the largest holes in the foam of the bores and the swash (m); the mark on the sand has holes a third as big. Real swash lace: 0.1-0.4 m. The rest of its look is Shoreline > Foam's , whose Look sets this too.
Front Line, Front Line Width	The solid white line at the front of the uprush, and its width (m) as the wave starts up the beach (40 % of it at the top).
Bore Clumps, Clumps Width	White water behind the front, and how far behind it reaches (m). Bore Clumps (times the Intensity) is also the white water of the bores rolling in over the surf zone, or of the surf on the crests with the Bore off. It appears as the wave brings it (rolling in with the bore) and lasts the bores' White Water time (Bore settings); the foam builds up over a short distance behind the front, so its leading edge is soft.
Lace, Lace Time	Foam lace left on the sheet, and how long it takes to break up (s).
Backwash Streaks, Streak Stretch	How much the draining backwash pulls the foam into streaks down the slope, and how much longer than wide it gets at most. It stretches from the top of the run-up, slowly at first, faster as the backwash speeds up.
Backwash Drift	How far the backwash carries the foam back down: 1 = along with the retreating front, 0 = it stays where the uprush left it and breaks up there.
Bore Carry, Absorption	In the surf zone, how far the passing bores carry the foam on the water towards the beach (1 = with the water they move), and how far ahead of a bore its face draws in and swallows the old foam (m). The foam behind a bore is new foam it brings, building up just behind its front, and only where the waves have broken (shallower than Breaking Depth).
Swash Mark, Swash Mark Time	The thin line of foam each wave leaves on the sand at its highest reach, and how long it stays (s): a layer or two of bubbles, a grey-white veil the sand shows through, that pops into holes, in clumps, as it dries.

The foam is the water's own (**Shoreline > Foam**): a layer of bubbles around holes of clear water, its pattern repeating only every 4096 holes, out of sight. The white water of the bores and of the front is fresh: thick, bright and finely holed, with milky aerated water behind the bores (**Underwater Bubbles**). The lace it leaves is old: thin, with round holes the water and the sand show through. Less foam opens its holes rather than fading it: as the lace ages, as the Intensity lowers it, and as the backwash soaks into the sand (**Infiltration**), the holes grow and merge and the last clumps go. The foam is carried by the water, up the beach and back down, and lies on top of the water and the wet sand, so the white line stays at the very front of the thin tip of the swash.

Wet Sand (switch)	Use
Intensity	Overall strength of the wet sand.
Darkening	How much darker wet sand is: the sand the water has left, and the sand seen through the swash's water (as wet as it is there; a few centimetres of water barely tint it, and the terrain is drawn dry).
Wetting Time	How long the sand takes to get wet once the water reaches it (s): the wetness comes in gradually where the thin tip of the water shows the sand.
Drying Time	How long the sand takes to dry once the water has left it (s), down to the Persistent Wetness.
Edge Softness	Width over which the wetness fades out below the highest point each wave reached (m): a soft upper edge that follows the shape of the water.
Persistent Wetness	Dampness the sand most waves reach keeps however long it has dried (fading out towards the top of the beach). 0 = it dries completely between waves.
Reflection	Reflection on the wet sand.
Sheen, Sheen Time	Shine of the water still standing on the sand just after the backwash, and how long it lasts (s). It reflects the scene like the water it comes from.

The wetness comes from the last four waves: each wets the sand its water covered, from the moment it got there, and starts drying from the moment it left, so it never switches on or off at once.

The swash is drawn per pixel on the ground as it is actually rendered (terrain, pebbles, rocks, a character's feet): over beaches the water mesh is lifted just above the highest water, and each pixel decides between the swash sheet, the sea, wet sand and dry sand. It is visual: the gameplay queries and buoyancy see the still water level there. Like the waves, it depends only on the position and the time, so it is identical on every machine.

Spray Particles (Shoreline > Swash > Spray Particles; on by default on a new Ocean) throws spray droplets from the bores near the camera, as they roll in over the surf zone and as they climb the beach, using the same model to find the fronts, in Play mode around the Main Camera, and outside Play mode around the Scene view's camera (**Preview In Edit Mode**; it shows as the Scene view redraws: turn on **Always Refresh** to see it move). It says why no spray is thrown, if so (no baked data, waves, shoreline, swash, bores or the shore's foam off, no Main Camera), and shows the droplets alive. It is kept in a **Shore Splashes** component that the switch adds to the water when needed; the component is hidden (everything is set in the water's Inspector) and is removed with the water. From a script: `GetComponent<ShoreSplashes>()` . Ranges (lowest, highest) have a slider; each droplet takes a random value in them.

Spray Particles	Settings
Appearance	Size (m, mostly small), Size Over Lifetime (curve), Color (tint), Opacity , Color Over Lifetime (gradient: fades in, then out), Stretch (off by default: on, the droplets stretch into streaks lined up with their motion, Stretch With Speed , Stretch Length , and Motion > Rotation does not apply).
Emission	The droplets come out of the foam's patches, only where the water is white as drawn (with the foam's Texture pattern, out of the procedural pattern of the same amount): without the shore's foam (Shoreline > Foam off), no spray. Rate (droplets per second along 100 m of front close to the camera, at full strength, where the foam is dense: the bores rolling in over the surf zone, a bore at least as tall as when it reaches the beach, starting on the water as it is there, waves included, and the bores climbing the beach, as the bore reaches it; less while a bore forms, as it collapses and where the foam thins), Distance from the camera, Half Rate Distance (the rate halves every this many metres away from the camera: more spray close by, less in the back, where it would only add up to a haze of small droplets and use up the Max Particles; 0 = the same rate up to the Distance), Max Particles , Burst (droplets thrown together), Min Bore Strength (the bore collapses as it climbs: no spray below it), Spread Along the front and Spread Behind it (m), Launch Height above the water (in bore heights).
Motion	Throw Height (m: how high the droplets are thrown at an Upward Speed of 1 by a bore at full strength, whatever the height of the bore; one collapsing up the beach throws them lower), Upward Speed and Forward Speed (times the speed reaching the Throw Height, and times the speed of the front), Max Forward Speed , Scatter (random sideways speed), Gravity , Drag , Lifetime (s, at most: a droplet that sinks into the sand, twice as deep as the largest Size so it is out of sight, or falls back into the water it was thrown from, ends there), Rotation (degrees, least and most: each droplet faces the camera, turned at random between the two; -180 to 180 every way). Scenes saved with Rotation under Appearance keep it; the former Height , in bore heights, cannot be kept in metres: set the Throw Height again.
Material	The droplets' material: texture, base colour, and Soft Distance where they meet the scene.

The spray is tested against the opaque scene only, so the lifted water mesh never hides it. From under the water it is not drawn: the droplets above the surface would show through it. From a script, call `ApplySettings()` after changing the curves, gradient, gravity, drag or stretch, and `Clear()` to remove every droplet at once (a camera cut).

9. Floating objects and boats

Add **Floating Body** to an object with a Rigidbody and colliders (or use **Make Selected Float**). It floats on the water it is in, or on the one set in **Water**. The volume of its colliders is sampled on a grid. At each point under the surface, every physics step applies:

- buoyancy: water density x g x submerged volume (Archimedes);
- drag against the motion relative to the water, so waves and currents push floating objects the way real ones do.

Setting	Use
Water	The water to float on. Empty = whatever water the body is in (river, lake or ocean).
Samples Per Axis	Sampling grid per side of the colliders' box. More points = more precise pitch and roll, higher cost.
Water Density	1025 = sea water, 1000 = fresh water.
Mass From Density / Density	Sets the Rigidbody mass from this density and the colliders' volume. 500 = light wood, 900 = ice; above the water density the object sinks. Turn it off to keep your own mass.
Water Drag / Water Angular Drag	How strongly the water damps motion and rotation.
Drag Per Axis	Drag multiplier along the object's local axes. Hulls: (1, 1, 0.1), so they glide forward and resist sliding sideways.
Make Waves / Wave Strength	Pushes the water where the body moves (needs a Water Interaction, see section 10).

Tips:

- Objects float the way their shape and weight make them float: a uniform tall cylinder lies on its side, and a light cube floats edge up. To keep something upright, lower its centre of mass (Rigidbody > Automatic Center Of Mass off, Center Of Mass below the middle), as ballast does on real buoys and boats.
- Box, sphere and capsule colliders are sampled exactly. Mesh colliders must be convex to be sampled precisely (a non-convex mesh collider counts as its full box).
- Call `BuildSamplePoints()` after changing the colliders at runtime.

Demo scripts (in `Demo/Scripts`, free to reuse):

Component	Use
Boat Controller	Drives a floating boat: W S throttle, A D rudder, or set <code>throttle</code> and <code>rudder</code> from code (Read Input off). The propeller only pushes while under water, and the rudder only steers when the boat moves.
Boat Circle Autopilot	Sails a Boat Controller around a circle (Centre, Radius, Throttle). Disable it to drive by hand again.
Demo Camera	The fly-through camera of the demo, with viewpoints and a key to take the helm.
Fish School	A school of fish, drawn with GPU instancing. It roams a box, below the water's surface and above the ground. It follows the water's clock, so it swims the same way on every run (and in recorded videos).
Ship Sails	A ship's set sails and flags, shaped every frame by the wind (in Edit mode too): they fill, luff, ripple in gusts and go aback against the mast when the wind is ahead. The wind is the water's (Wind Source : the ship's water, or a fixed wind), less the ship's own velocity, kept between Minimum Wind and Maximum Wind . The mesh is made at runtime and never saved with the scene; in Play mode the sails move every Far Interval frames beyond Far Distance , and not at all when no camera sees them.
Night Fires	Lights a group of fires (Torch Fire) as the sun sets, one after another, and puts them out at dawn: their lights, flickering, their coals, and their flames and smoke, emitted into two shared particle systems and leaning with the water's wind. Mode : Follow The Sun, Always Lit, Always Out. The scene is saved with the fires out (lights off by day cost nothing).
Torch Fire	One fire of a Night Fires group: where its flames and smoke rise, how big and how many, its unshadowed point light (Light Intensity, Flicker), how far into the dusk it lights (Light At), and the share of its flames it keeps by day (Day Flames : a cooking fire that smoulders).
Seagull Flock	Gulls that circle over their Hotspots , glide, flap, land and ride the waves, stand on Perches , keep clear of Obstacles , take off when a Disturber (a ship) comes close, and roost at night. Drawn with GPU instancing (a body and two wings each); simulated in Play mode on a fixed step, seeded (Seed : the same flight every run).
Crab Colony	Crabs that walk sideways on the sand around their burrows, stop and raise their claws as the camera comes near, and run into a burrow, into the sea or into the sand when it comes closer, coming back out once it has gone. They walk on a baked mask of the beach (the dev tools bake it from the terrain, the trees and the props). Drawn with GPU instancing, simulated in Play mode near the camera only.
Terrain Detail Lightmap	On a terrain: gives its detail meshes (grass, ferns) its baked lightmap where they grow, so their ambient light is the ground's under them (the canopy's shade, the light it bounces); their sun shadows beyond the realtime shadow distance still come from the light probes. The terrain draws them with GPU instancing, which takes no lightmap; their material must use Island Foliage with Terrain's Lightmap on. Without a lightmap, or with Subtractive lighting, they take the light probes.
Network Water Clock	Shares a server clock between players (see section 11).

10. Interaction: ripples and wakes

A **Water Interaction** component simulates a square of water around the camera, or around its **Target**. Creating an Ocean adds one. Floating Bodies (with **Make Waves**) and **Water Interactor** spheres push the water, and the resulting waves are added to every water surface inside the square.

- The waves are evolved exactly: every wavelength travels at its true deep-water speed. Objects dropped in the water make rings, and moving boats leave a real Kelvin wake at any speed.
- Only what objects do relative to the water makes waves. An object riding the swell makes none; one moving through the water, bobbing or dropped in does.
- Foam appears where objects churn the water fast (a planing hull, a splash) and where the waves get steep.

Setting	Use
Target	Centre of the simulated square. Empty = the main camera.
Resolution / Size	Texels and metres per side of the square. Finer ripples need more texels per metre.
Damping	How fast the waves die out.
Strength	Height of the waves objects make. 1 = the water they really displace.
Foam Lifetime, Foam From Objects, Foam From Steepness	Foam of the interaction waves.

Water Interactor is a sphere (**Radius, Strength**) that pushes the water without floating: a swimmer, a paddle, a character's legs, a projectile. **Tools > Havoila Studio > Drop-In Water > Selection > Make Selected Push the Water** adds one sized to the object. In Play mode the Water Interaction Inspector shows the simulated square and how many objects push the water.

The interaction waves are visual: gameplay queries and buoyancy use the deterministic waves only.

11. Gameplay: queries and multiplayer

Querying the water

Every water body (Ocean, Lake, River) answers the same queries. The waves are read from a CPU copy computed for the current frame on the worker threads, so the queries answer immediately and cost microseconds. The copy is only computed while something queries it: the first query after a while (half a second without any) computes it then. They work in FixedUpdate (the waves at the physics step's time, computed two steps ahead) and on a server with no GPU.

Call	Returns
<code>water.SampleHeight(position)</code>	height of the surface above/below a point
<code>water.IsUnderwater(position)</code>	true below the surface
<code>water.SampleSurface(position)</code>	height, normal and velocity of the water (orbital motion, river flow)
<code>water.ScheduleHeightQueries(positions, heights)</code>	the same for thousands of points, in a Burst job
<code>water.ScheduleSurfaceQueries(positions, samples)</code>	height, normal, velocity in a Burst job
<code>water.GetUnderwaterState(camera)</code>	above water / crossing the surface / underwater
<code>ocean.GetWaveField()</code>	raw access for your own Burst jobs (call <code>ocean.RegisterReader(handle)</code>)
<code>river.GetRiverField()</code> , <code>river.Locate(position)</code>	river data for Burst jobs, and where a point is across the river; <code>field.Depth(location)</code> : how deep the water is there (its baked bed, or the bed it assumes; 0 over a bank or a rock)
<code>river.Data</code> , <code>RiverBaker.Bake(river)</code> (Editor)	the ground baked under the river (RiverData), and baking it from a script
<code>WaterBody.Find(position)</code> , <code>WaterBody.TrySampleSurface(position, out sample)</code>	the water at a position (river, lake or ocean)
<code>ocean.ResetFoam()</code> , <code>spray.Clear()</code>	nothing: they forget the whitecaps' trail and remove the spray's droplets, for a camera cut or a sudden change of weather

```
using HavoilaStudio.DropIn.Water;

if (WaterBody.TrySampleSurface(transform.position, out var water) && transform.position.y < water.height)
    Debug.Log($"Under water, current {water.velocity}");
```

At a river's mouth or source the queries blend as the surface does: the current fades out into the lake or sea and their waves come back, with no step where the river ends.

Where a body has no water, its queries return a height of -1 000 000 (below any position): beside a river's banks, and outside a lake's box and basin.

Complete the jobs within the frame. `Ocean.Active` is the ocean enabled last. In `FixedUpdate` the queries return the waves at the time of the physics step, not of the frame, so physics gives the same result at any frame rate.

Multiplayer

The waves are a function of the settings, the seed and the time: every client computes the same sea from the same time, so nothing about the water needs to be sent. Share the clock:

```
WaterClock.TimeSource = () => NetworkManager.Singleton.ServerTime.Time; // Netcode for GameObjects
```

The **Network Water Clock** demo component does this with smoothing. Call `Synchronize(serverTime)` when your transport gives you the server time, and the waves catch up without jumping.

For bit-identical physics between machines (lockstep, rollback), step the physics at the same times on every machine and make `WaterClock.Now` return the time of the step (e.g. `tick * tickInterval`). The waves repeat exactly after **Loop Period** seconds, so a long-running server keeps full float precision. A server built with `-nographics` still computes the waves for queries and buoyancy.

12. Tools menu and debug views

Tools > Havoila Studio > Drop-In Water >	Does
Create > Ocean / Lake / River / Water Interaction	Creates the water. Lakes and rivers are placed at the centre of the Scene view. It warns before a second ocean and selects the existing Water Interaction if there is one.
Selection > Make Selected Float	Adds a Rigidbody and a Floating Body (and a Box Collider if there is no collider).
Selection > Make Selected Push the Water	Adds a Water Interactor sized to the object.
Bake Shoreline of Selected / Bake All Shorelines in Scene	Bakes the terrain under the ocean and lakes (section 8).
Bake River Bed of Selected / Bake All Water in Scene	Bakes the ground under the selected rivers; or every shoreline and river bed of the open scenes, with a summary.
Fit Selected River to Terrain	Fits each river point's width to the banks.
Debug View > ...	Replaces the shading of every water surface with a diagnostic view (below). The same views are in the menu of the Ocean, Lake and River components (right click on the component's header, or its : button).
Documentation / Online Documentation / Contact Support	This guide, the web version, and an email to support.

The debug views show what the refraction does. From a script: `WaterDebug.View = WaterDebugView.RefractionScale`.

The refraction follows the bent view ray to the bottom. Where that point is hidden (behind an object in front of the water, or off screen), its distance is estimated, so the absorption stays continuous; the colour is then read closer to the pixel, smoothly, from what the screen shows under the water.

- **Refraction Scale.** How much of the refraction offset the colour lookup uses. White = all, darker = pulled towards the pixel near a screen edge or an object in front of the water.
- **Refraction Offset.** Red and green show the offset of the colour lookup (0.5 = none).
- **Refraction Grid.** A checkerboard seen through the refraction. It should bend smoothly, never shatter.
- **Refraction Search.** How much an object in front of the water limits the colour lookup.
- **Refraction Foreground.** Where the bottom the ray reaches is hidden: white = behind an object in front of the water, grey = off screen (both estimated).
- **Refraction Distance.** Length of the light's path through the water (white = 16 m). It should stay continuous around objects.
- **Swash.** On beaches: red = swash water, green = wet sand, blue = foam; dark blue = the sea elsewhere.
- **River Flow.** On rivers, their flow map: red = the speed of the pattern carried by the flow (1 = 5 m/s), green = white water (1 = 8 m/s), blue = the depth at the water's edge (1 = 4 m; full where it is unknown).
- **Foam.** Every foam (the water's, the shore's, the swash's): red = how much of the pixel it covers, green = its thickness (1 = 40 bubble layers), blue = the air under it (Underwater Bubbles).

13. Performance

Cost depends on the screen coverage of the water, the simulation settings, the number of water bodies and cameras, and whether the camera is under water. Profile in your own scene and on your target hardware; the Profiler and the Frame Debugger show the water's work as `DropInWater.WaveSimulation`, `DropInWater.CausticRipples`, `DropInWater.Mask` and

Cost	Setting
Wave simulation (GPU)	One FFT per cascade and per Ocean or Lake: Quality > Resolution (256 by default) and Cascade Count (4). Small, calm lakes can often use 128 and 3 cascades. With Caustics off or Wave Lensing at 0, the pass that prepares the caustics' broad light is skipped too.
Caustics (GPU)	One small FFT per Ocean, Lake or River for the caustics' ripple net: Quality > Caustics Resolution . 512 (the Ocean's default): about 0.1-0.15 ms and 15 MB; 256 (Lake): about 0.04 ms and 4 MB; 128 (River): about 0.01 ms and 1 MB. Off, or Caustics off, costs nothing. Per water pixel that sees the bottom: 2 texture reads of that net, 4 with Dispersion and an oblique sun (a river: 4). Changing Cell Variety, Alignment or the wind direction rebuilds the net on the CPU (a few milliseconds).
Gameplay copy (CPU)	Quality > Gameplay Resolution (64 by default), computed in a Burst job. It sets the precision of queries and buoyancy.
Surface mesh (GPU)	Quality > Grid Resolution and Levels set the vertex count; Vertex Spacing sets the detail near the camera. Levels beyond the camera's far clipping plane are not drawn.
Reflections (GPU)	SSR Steps per pixel. Turn Screen Space Reflections off for probe and sky reflections only.
Sky reflection (GPU)	With Skybox on: the skybox drawn into a 128 x 128 cubemap and filtered, with its ambient light, only when it changes (at most ten times a second). Nothing while the sky stays the same. With a sky that hands over its own environment, only the ambient light (a small cubemap, ten times a second).
Underwater (GPU)	Only while a camera is under water: Light Shaft Samples per pixel, or Light Shafts off.
Sky fog (GPU)	Only with a sky that publishes its fog: one texture read per water pixel, two more where a screen-space reflection hits.
Point & spot lights (GPU)	Per water pixel, the lights URP gives it (Forward+: those around it); under water, every visible light whose range the view ray crosses, in closed form: 2 evaluations per light, 8 for a spot light or a light with a cookie (its beam). With Shadows in the Water : a shadow lookup at the point under the surface per light, and under water Beam Shadow Samples shadow lookups along the beam of each light that casts shadows, per pixel. Switch Water > Point & Spot Lights off if the water does not need them.
Interaction (GPU)	Resolution of the simulated square (256 by default). Remove the component if nothing pushes the water.
Buoyancy (CPU)	Samples Per Axis cubed points per floating body, in one Burst job per water body.
Rivers	The river mesh (Segment Length , Cross Segments) and the ripple simulation of each river (on the GPU only: queries follow the flow, not the ripples). Where a river blends into a lake or the sea, both are drawn over its first or last blend metres, and the lake or sea reads a small map of the river (128 x 128, made when the river changes) near its end.
Foam (GPU)	Only where there is foam: water without any costs a few instructions. Its holes are computed (no texture read with the Procedural pattern; with Texture, one read per size of holes, two more for the Relief): up to three sizes up close, set by Quality > Foam Detail (Medium draws two, Low one; the River's is in its Foam section), fewer further away, where each fades into its average, and only an average far away. Bubbles at 0 saves a pass within a few metres.
Swash (GPU)	Only on beaches, within the swash zone: a depth read, a few noise reads, and the foam of its parts where they have some (usually one or two per pixel). Switch the Swash off to remove it.
Spray Particles (CPU)	A few hundred model evaluations per second and one particle system; Rate and Max Particles .

14. Limitations

- URP only. Compute shaders are required: mobile, WebGL and XR are not supported in this version.
- One ocean per scene. Lakes are boxes, rotated around Y only; with baked data their water keeps to their basin inside it.
- A lake's basin is found from the baked terrain: a rim made of meshes is not seen (the lake fills its box there), and a gap in the rim lower than the water lets the lake through.
- Interaction waves are visual: they do not move floating bodies and are not returned by the queries.
- Water is not removed inside hulls: an open boat that sits low shows water inside it below the waterline.
- Rivers have no dedicated waterfall rendering: steep drops are drawn as fast white water.
- A river's flow follows its bed on a grid in its own space (rows along it, columns across): a rock narrower than about two texels (1 m at the default Texel Size) is not seen. The water is steered around rocks and leaves slow, still water in their lee: it does not swirl in eddies there.
- A river's bed only lowers the ground: it builds no levee where the water stands above the ground beside it. It carves terrains only (not meshes), never one whose TerrainData is shared by two terrains or read-only, nor in Prefab Mode. Ground cut below a terrain's lowest height is cut at that height.
- Rivers blend into lakes and the sea, not into other rivers. Up to four river ends blend into one lake or sea at a time: the nearest to the camera.
- Shorelines are baked. The bake follows terrain sculpting done in the Editor with its water's scene open (section 8), but not terrains added, moved or resized (it says so: bake again), terrains changed at runtime, nor moved colliders in a bake that took them. Only terrains are baked by the menu items.
- The swash and the bores are visual: queries and buoyancy do not see them. Over beaches the water mesh is lifted above the highest swash water, but writes a depth close to what it shows there (the water, or the wet sand) rather than its own: transparent objects drawn after the water, very close to its surface or to the sand in the swash zone, can still be hidden (the included spray is not).
- Refraction and screen-space reflections only show what is on screen, and transparent objects behind the water are not refracted. Past the sides of the screen, the reflections take the scene as going on as the edge of the screen shows it (a shore or a tree line running out of the screen stays reflected, stretched), fading out to the reflection probes and the sky over a third of the screen's width.

15. Troubleshooting

- **The water is pink.** URP is not the active render pipeline (**Project Settings > Graphics and Quality**).
- **The water reflects another sky** (a bright day sky at dusk, or the sky before you changed it). With **Water > Reflections > Skybox** off, the water reflects the environment reflection generated in the Lighting window, which keeps the sky and sun it was made with: turn Skybox on, or press **Generate Lighting** again. The ambient light of the scene is generated the same way: with Skybox on the water takes its own from the sky, but the other objects (and the ground seen through shallow water) keep the generated one until you generate the lighting again.
- **The water is black or reflects black.** With Skybox off, the scene has no environment reflection. Open **Window > Rendering > Lighting** and press **Generate Lighting**, with Environment Reflections set to Skybox.
- **The sea shows on dry land below sea level, or in a valley.** Bake the shoreline (section 8).
- **The swash runs up inside a beach, or spray comes out of the sand.** The terrain changed since the bake while the water's scene was not open: press **Update From Terrain** in the water's Inspector (the Shoreline section says "terrain changed"), or bake again.
- **A calm strip of sea, edged with shoreline, runs out to the horizon.** The ocean's baked area ends on dry ground there, and the water keeps that ground beyond it (section 8). Press **Bake Shoreline** again (it bakes past the terrains), or, with an area of your own, bake a larger one with all its edges under water: the Console says where it ended on dry ground.
- **The waves are too big on a small lake.** Keep **Fetch From Size** on, or lower the wind speed.
- **An object sinks, jumps or spins.** Check that its density is below the water density, that its mesh colliders are convex, and that the colliders match its shape. Call `BuildSamplePoints()` after changing colliders at runtime. To keep it upright, lower the centre of mass.

- **There are no ripples or wakes.** The scene needs a **Water Interaction**, the object must be inside its square (around the camera or Target), and it needs **Make Waves** (Floating Body) or a **Water Interactor**.
- **The seabed shimmers through the water when the camera moves.** Raise **Refraction Smoothing** on the material (default 8): the refraction then follows the larger waves only.
- **Shadows of tall objects disappear close to the camera.** Piers, masts or cliffs lose part of their shadow near the camera, cut along a curved line. This is URP's default **Shadow Near Plane** of 0.2 on the directional light: raise **Light > Shadows > Near Plane** to 3 - 10.
- **A lake shows water beyond its basin.** Set its **Shape** to Basin and bake the shoreline (section 8). If it still does, the Lake section says why: the lake reaches the edge of its box (a gap in the rim lower than the water: enlarge the box and bake again, or raise the rim there), a river's valley too wide to dam, or the object over dry ground. With Shape = Box, shrink **Size**, or raise the ground there.
- **A river flows the wrong way.** Press **Reverse Flow** in the River Inspector.
- **A river spills over its banks.** Its Inspector says between which points the water stands above the ground beside it: lower those points (or raise the ground there). Carve Bed only digs.
- **A river's bed did not follow.** Its Inspector says why (a TerrainData shared by two terrains, or read-only). Press **Recarve** to dig the whole bed again.
- **A river's ripples and foam stretch into streaks or arcs.** Shorten its **Flow Cycle** (material, Flow section): the Flow section says how much of the river its pattern follows exactly.
- **A river turns white everywhere, or nowhere.** Raise or lower **Rapids Speed** (material): the Flow section says how much of the water is white. Bake its bed: shallow, fast water and rocks then foam, deep water does not.
- **A river ends in a hard line in a lake or the sea.** Turn on **Mouth and Source > Blend With Water**, and make its first or last point lie over their water, from 1.5 m under their level to 1 m above it: its last **Mouth Blend Length** (first **Source Blend Length**) metres then fade into them. A river ending higher above them (a waterfall) is left as it is.

16. Support

Havoila Studio · support@havoila.com · havoila.com

Online documentation: havoila.com/assets/drop-in-water/docs/

Please include your Unity and URP versions, the target platform, a screenshot of the Inspector and any Console errors.

Tools > Havoila Studio > Drop-In Water > Contact Support opens an email with the version filled in.

17. Version history

- **1.0.0:** first release.