2D

E R A G A N X

2D Water Reflection PRO

Only 2D Reflections. No Physics

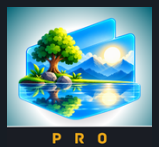


E R A G A N X | UNITY PACKAGE GUIDE

The complete guide from the beginning

Scene setup, every Reflection Surface setting, custom shapes, presets and C# usage - step by step.

ENGLISH EDITION | SCREENSHOT GUIDE | C# EXAMPLES



Source target: Unity 6000.3.13f1 | URP 17.3.0

11 October 2026

START HERE

How to use this guide

Follow the first-scene steps, then tune the Reflection Surface Inspector. Each screenshot uses the interface supplied by ERAGANX; values shown are examples. C# examples use the actual public API in the current source.

Reading path - click to jump	Pages
Foundation, installation and the first scene	03 - 07
Inspector layout, shape, modes and filtering	08 - 13
Waves, ripples and reflection appearance	14 - 19
Quality, pixel art, caching and advanced settings	20 - 25
Built-in presets and custom preset assets	26 - 29
Official samples and demo controls	30 - 33
C# setup and complete runtime examples	34 - 37
API quick reference	38 - 39
Troubleshooting and the final scene checklist	40 - 42
FAQ, glossary, sources and version scope	43 - 44

Choose your starting point

New scene: start at page 4 and create one Planar surface. **Existing scene:** check filtering and capture geometry before adding effects. **Performance:** read Quality and Optimization. **Custom shapes:** start at page 10. **Gameplay controls:** go to the C# section.

> **Saving changes**
> Save scene and prefab settings in Edit Mode. Ordinary Inspector changes made in Play Mode are lost when Play Mode ends. Collapsing a Settings foldout does not disable its section.

01 - FOUNDATION

What this package does



ERAGANX marketing illustration. The actual sample output is shown on page 30.

2D Water Reflection Pro is a rendering tool with shaders, runtime components and Editor controls. It captures or mirrors selected 2D scene content onto a flat reflection surface. The package includes demonstration artwork; you can use your own sprites, Tilemaps and compatible effects.

Setting	Purpose / action
Scene reflections	Five modes: Planar, Lightweight Sprite, Cached, Hybrid and Mirror.
Selection	Included/excluded object layers, exact renderer selection and sorting-layer filters.
Surface silhouette	Rectangle by default, or a supplied sprite alpha mask with circles, triangles, irregular edges and holes.
Appearance	Directional waves, radial visual ripples, colour/tint, brightness, saturation, offset, blur, fades and soft edges.
Workflow	Branded Inspector, Scene View handles, presets, runtime API and ten sample scenes.

Scope
Designed for compatible orthographic 2D scenes, including platformers, adventure scenes, menus and decorative puddles or mirrors. It is not limited to side-scrolling gameplay. It supplies visual reflections only: no water physics, buoyancy, collision ripples or fluid simulation.

01 - INSTALLATION

01. Prepare a compatible Unity project

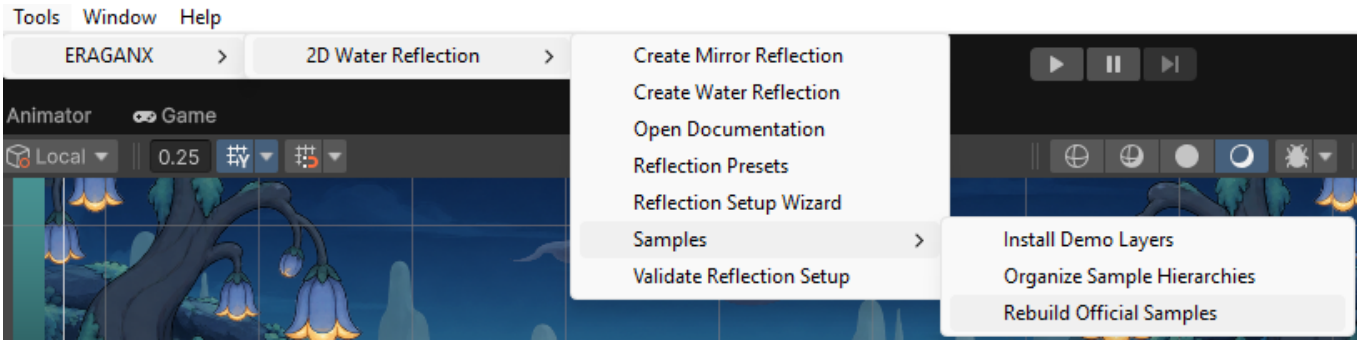
Setting	Purpose / action
Source target	Unity 6000.3.13f1 with Universal Render Pipeline 17.3.0. These are the versions inspected for this guide.
Render pipeline	URP with a 2D Renderer. Built-in RP and HDRP are not supported by the current capture backend.
Camera	Orthographic Base camera facing +Z. No perspective camera, X/Y camera tilt, XR or camera stack.
Surface transform	Positive, orthogonal XY scale. No negative scale, shear or X/Y tilt. Z rotation is supported with the same flat-plane geometry.
Shaders	Shader model 3 support. Retain the bundled Resources materials and shader includes.
Your script asmdef	Reference ERAGANX.WaterReflection2D. Add the Samples assembly only when using sample-specific classes.

1. Create or open a Unity 6 URP 2D project. Check the active URP asset in Project Settings > Graphics and Quality, and ensure its renderer is a 2D Renderer.
2. Import the package with its Runtime, Editor, Shaders, Resources, Presets and Samples folders intact. Wait for compilation to finish and resolve Console errors first.
3. Open one official sample in the package Samples/Scenes folder. Demo layers use project-specific numeric indices: in a new project, use **Rebuild Official Samples** once to install free named layers and remap the bundled samples.
4. Create a separate scene for your game. Keep custom scenes and modified presets outside the bundled sample folders so a sample rebuild cannot overwrite them.

> **A clear first test**
> Start with a stationary camera and one unlit sprite above the waterline. Verify reflection geometry before adding lights, effects, camera follow or caches. Test a player build on your actual target platform before promising platform support.

02 - TOOLS

02. Find the package tools



Tools > ERAGANX > 2D Water Reflection. All entries belong to the Editor workflow.

Setting	Purpose / action
Create Water Reflection	Creates and selects a surface, connects an orthographic camera and ensures a reflection manager exists.
Create Mirror Reflection	Creates a Mirror-mode surface with full strength and opacity. Horizontal is the default orientation.
Open Documentation	Opens this English PDF in your system PDF application.
Reflection Presets	Selects Calm Lake. Missing built-in presets are created; existing assets are retained.
Reflection Setup Wizard	Choose the source camera and optional clean mirror, then create a surface.
Validate Reflection Setup	Checks loaded surfaces for supported configuration. It does not prove visible rendering.
Install Demo Layers	Adds the three named demo layers to free project slots. It does not remap existing sample objects.
Organize Sample Hierarchies	Groups official sample objects while preserving world transforms, settings and references.
Rebuild Official Samples	Recreates all ten bundled scenes and prefabs, installs demo layers and remaps objects to them.

Before rebuilding samples
Save custom copies of any edited official scene or prefab. Rebuild is an explicit authoring operation that replaces bundled sample content. The package does not install layers or rebuild scenes automatically on import.

02 - FIRST SCENE

03. Create your first water reflection

1. Place a SpriteRenderer above the intended waterline. Set the Main Camera to Orthographic with rotation (0, 0, 0); keep the sprite inside the camera view.
2. Choose **Tools > ERAGANX > 2D Water Reflection > Create Water Reflection**. The selected object has the Reflection Surface component. An ERAGANX Reflection Manager is created if needed.
3. Move the surface Transform to the waterline. Use positive unit scale for the first test. The default horizontal destination extends down from this position.
4. In Reflection, assign Source Camera and set Surface Size to (10, 3). In Reflection Mode, select Planar. Use Output sorting layer/order to place the reflected output above the water background and below any foreground bank.
5. In Capture Region, use X = -5, Y = 0, W = 10, H = 3. This covers source content from five units left to five units right, and zero to three units above the plane.
6. Include the source sprite's object layer. Exclude water backgrounds, shore decoration and anything you do not want mirrored. Begin with no object or sorting-layer restrictions.
7. Enable Editor Preview in Optimization, or enter Play Mode. Confirm that the sprite appears below the waterline. Keep Visual Effects off during this first geometry check.
8. Save the scene in Edit Mode. Add a separate water-bed sprite if you want an opaque coloured background under transparent parts of the reflection.

What you should see
The reflection follows the selected scene content, clipped to the destination rectangle. Transparent source areas remain transparent when Background Color has alpha 0. A surface does not create a player controller, collider or water texture.

02 - GEOMETRY

04. Understand the plane and destination

Capture Region describes source content. Surface Size describes the displayed destination. Both use the surface's local plane coordinates. Transform position places the reflection plane; Transform scale also changes its world size.

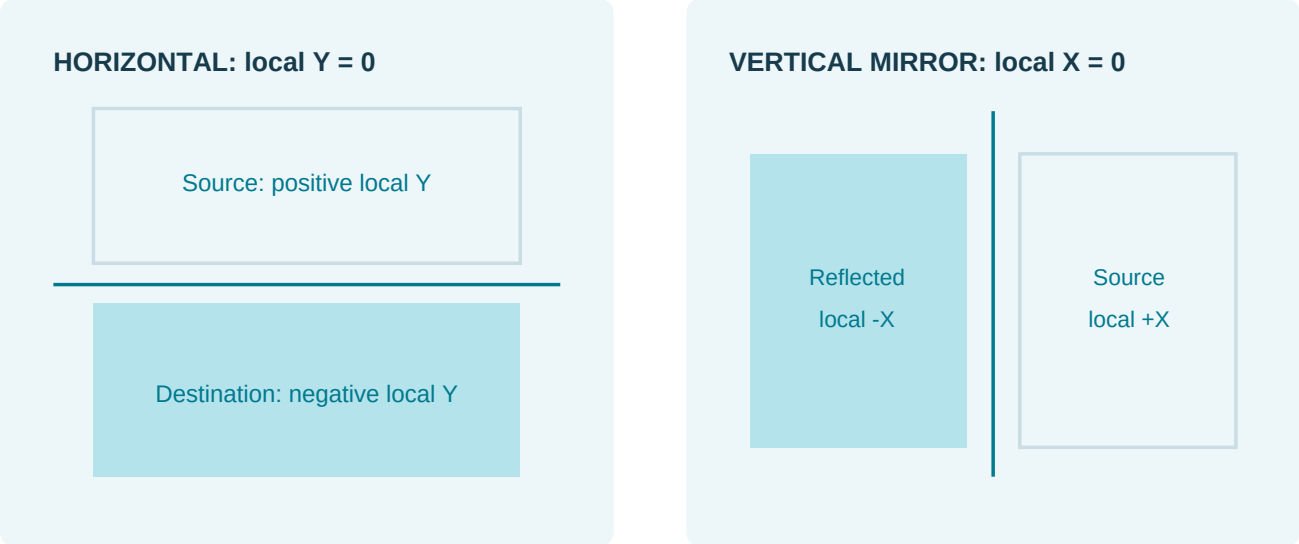


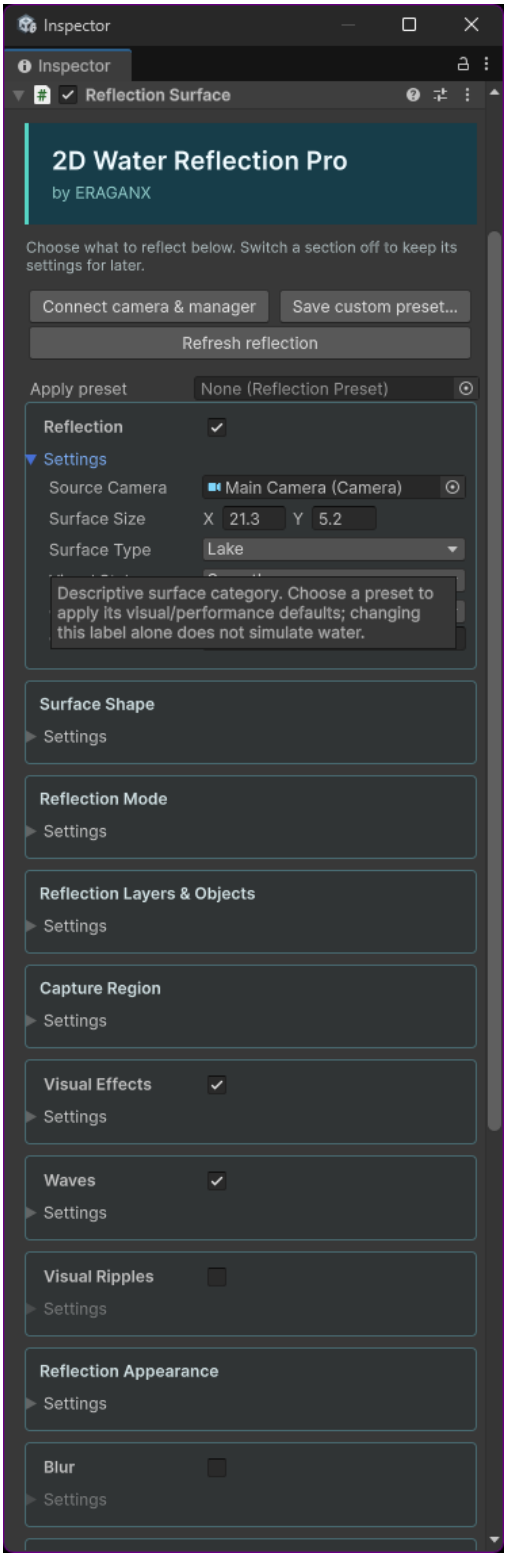
Diagram: geometry convention, not a rendered scene.

Setting	Purpose / action
Horizontal water / mirror	Surface width runs along local X; height extends down along local -Y. Source plane coordinates map directly to local (X, Y).
Vertical Mirror	Plane coordinates (tangent X, normal Y) map to local (Y, -X). The destination is on local -X; the source is on local +X.
Resizing	Use Surface Size or Scene View handles. Capture Region is edited separately; resizing the destination does not select a new source region automatically.
Custom alpha shape	Changes visible silhouette inside the destination. The reflection still uses one flat plane and rectangular capture/visibility bounds.

Avoid accidental stretching
Capture enough source content for the intended destination and framing. When using a shape sprite, Match sprite aspect preserves its silhouette proportions by adjusting destination height while keeping width.

03 - INSPECTOR

The complete Reflection Surface Inspector



Inspector overview. Detailed, readable section screenshots follow.

One component
Select the Reflection Surface object. The branded Inspector groups camera, shape, selection, effects and performance settings in one place.

Toolbar
Connect camera & manager finds an orthographic camera, ensures a manager and enables Editor Preview. Save custom preset stores current reusable settings. Refresh reflection requests a new capture.

Apply preset
Choose a Reflection Preset asset. Scene bindings and surface geometry remain intact.

Section checkbox
Turns that feature on or off and retains its values for later. Waves, ripples, blur, fade and soft edges also enable Visual Effects when switched on.

Settings foldout
Expands or collapses controls. Folding the card does not pause an enabled effect.

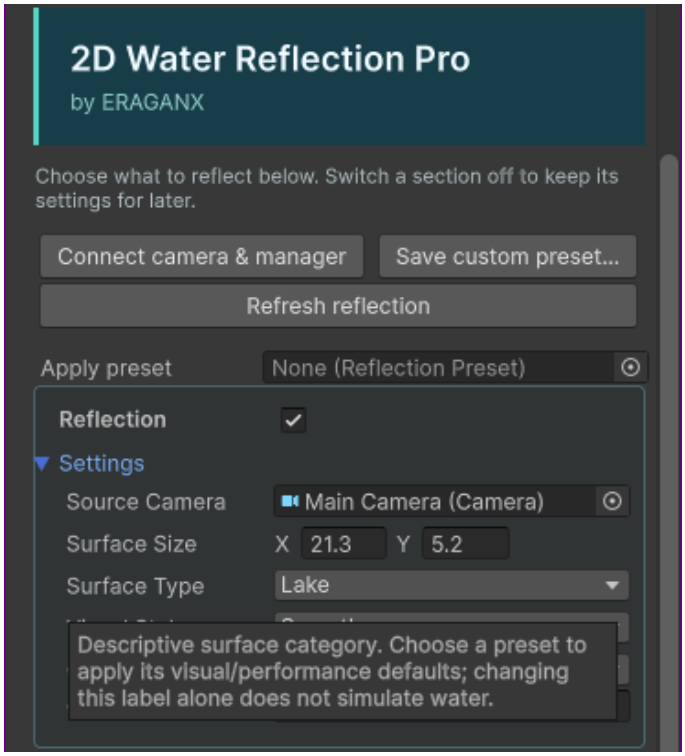
Tooltips and Undo
Hover fields, dropdowns, buttons and foldouts for help. Serialized editing supports Undo and multiple selected surfaces.

Scene View
Move and resize the surface using its handles. Keep a visible source-camera frustum when off-screen skipping is enabled.

A screenshot may show a hover tooltip over a field. That is Editor help, not an error. Inspector values are examples rather than required defaults.

03 - REFLECTION

05. Camera, size, style and sorting



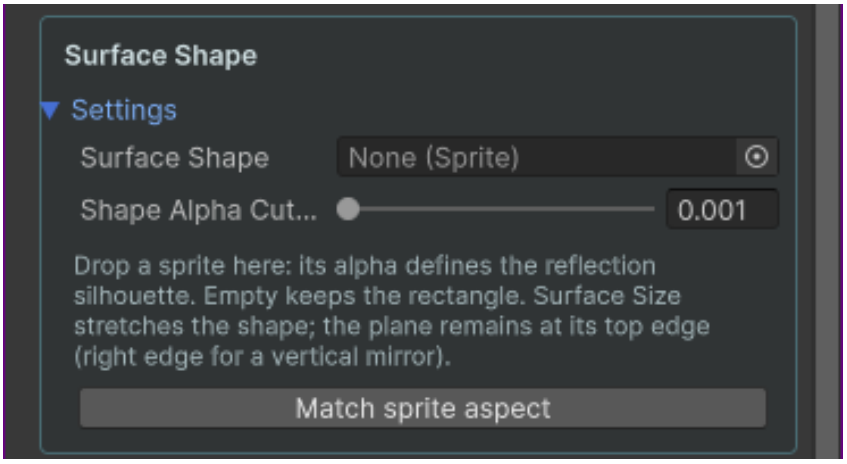
Reflection section and toolbar. The visible tooltip explains Surface Type.

Setting	Purpose / action
Reflection checkbox	Enables/disables this surface's output. Disabled sections keep their configured values.
Source Camera	The orthographic URP 2D Base camera whose scene content and view are used for reflection.
Surface Size	Destination width and depth in local plane units. Horizontal output extends below the plane; vertical mirror output extends to its left.
Surface Type	Descriptive category: Lake, River, Ocean, Puddle, Wet Floor, Mirror or Custom. Changing the label alone does not simulate water or apply a preset.
Visual Style	Smooth uses Bilinear capture filtering. Pixel Art uses Point filtering with the pixel settings described on pages 20-21.
Output sorting layer / order	Draw order of the displayed Planar, Cached, Hybrid or Mirror output. Lightweight preserves individual source sprites' sorting.

Layer and sorting are different
An object layer decides what can be captured. A sorting layer/order decides draw priority. If water is hidden behind a background sprite, fix output sorting rather than adding more included object layers.

03 - SURFACE SHAPE

06. Use your own sprite silhouette



Surface Shape accepts a Sprite. Its alpha defines the output boundary.

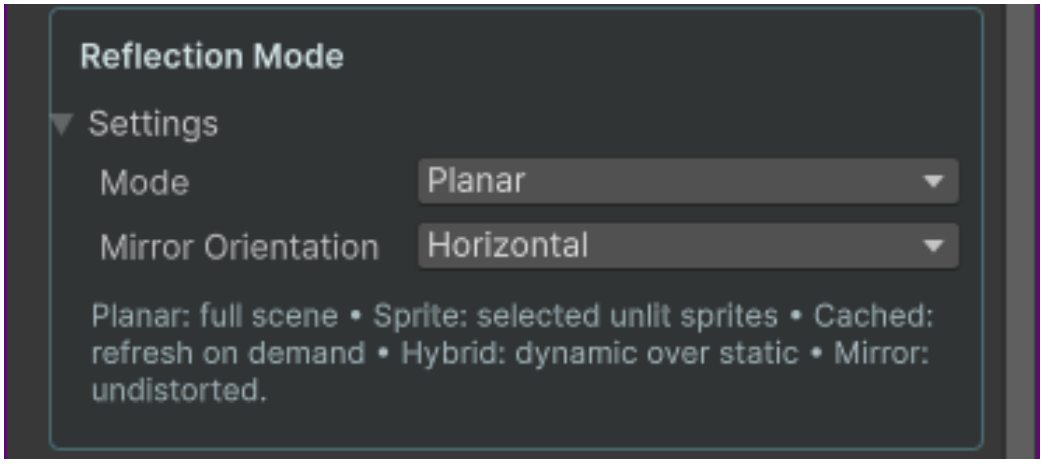
1. Prepare an RGBA PNG with transparent pixels outside the wanted shape. Import it as Sprite (2D and UI), retain alpha and avoid Tight Packing in a Sprite Atlas.
2. Drag the Sprite into **Surface Shape**. Its RGB colours are ignored; alpha masks the reflection. Leave it None for the standard rectangle.
3. Click **Match sprite aspect** to keep the shape's width/height proportions. Otherwise Surface Size stretches it to the destination frame.
4. Use **Shape Alpha Cutoff** to hide very faint alpha. A small value such as 0.001 preserves soft transitions; higher values remove more translucent pixels.

Setting	Purpose / action
Supported silhouettes	Circles, triangles, irregular boundaries, transparent holes and soft alpha edges. All five reflection modes can use the same mask.
Sprite placement	The full original sprite rectangle is normalized over the destination; the sprite pivot does not move the reflection plane.
Texture setup	An unpacked sprite or rectangular atlas packing is supported. Read/Write is not required. Point/Bilinear import filtering affects the alpha edge.
Plane and cost	The plane remains at the rectangle's top edge, or right edge for a vertical mirror. A small visible silhouette does not reduce the rectangular capture area.

No curved mirror surface
A circular mask makes a circular window onto a flat reflection. It does not generate curved normals, water geometry or physics. Tight-packed masks are rejected; rotated rectangular atlas and platform split-alpha formats need target-platform validation.

03 - REFLECTION MODES

07. Choose a reflection mode



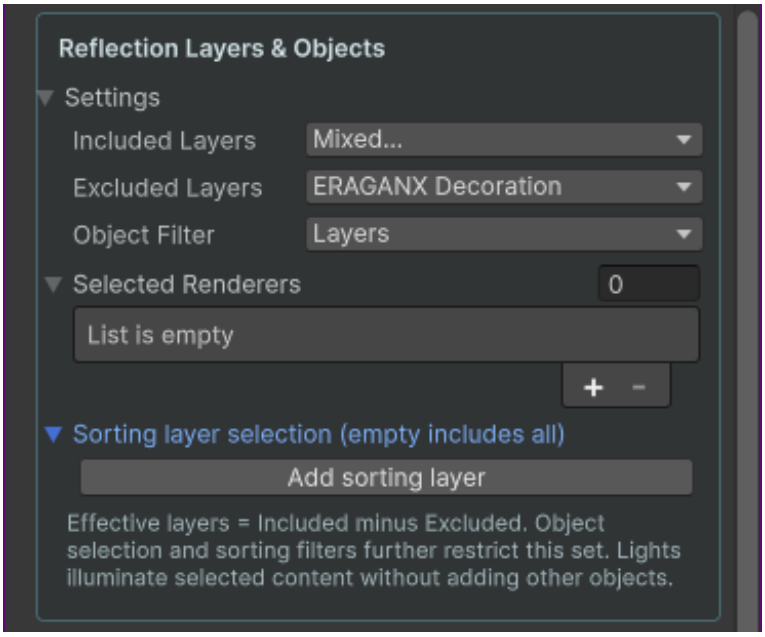
Mode and Mirror Orientation. In C#, Lightweight Sprite is ReflectionMode.LightweightSprite.

Setting	Purpose / action
Planar	Live scene capture. Best general starting point for SpriteRenderers, Tilemaps, compatible particles and native 2D lighting.
Lightweight Sprite	Mirrors selected Simple, unmasked, unskinned Sprite-Unlit-Default renderers without property blocks. No full scene camera/RenderTexture capture. Use Planar for lit/custom materials or complex sprites.
Cached	Stores a scene capture and reuses it until invalidated or refreshed. Suitable for static scenery. Moving characters and light changes can remain frozen.
Hybrid	Caches included static layers and refreshes the chosen Dynamic Layers. Dynamic content is composited over static content; interleaved transparent sorting needs Planar.
Mirror	Clean scene reflection with horizontal or vertical orientation. Waves and ripples are suppressed; appearance, blur and fade remain available if enabled.
Mirror Orientation	Horizontal reflects across local Y = 0. Vertical applies only to Mirror and reflects across local X = 0.

Choose by content
Use Planar when exact scene composition matters. Lightweight requires Object Filter = Include Selected and a nonempty Selected Renderers list of compatible sprites. Cached/Hybrid benefit depends on actual scene changes; no universal FPS or mobile performance guarantee is implied.

03 - SELECTION

08. Select reflected layers and objects



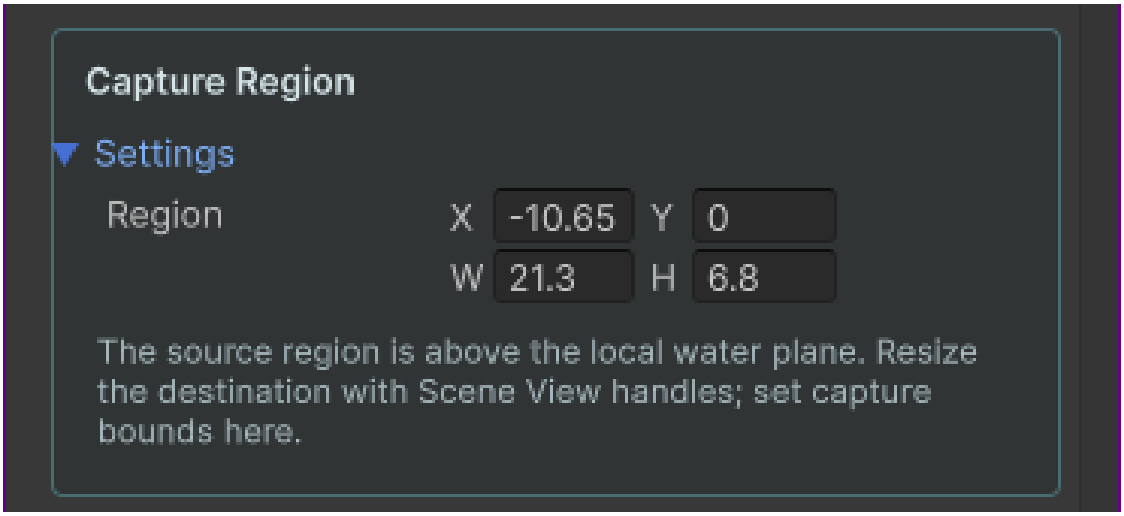
Layers & Objects. Mixed... can be the normal display for a multi-layer mask.

Setting	Purpose / action
Included Layers	Object layers allowed into the capture. Include only the scene content you need.
Excluded Layers	Removed even when also included. Effective layers = Included AND NOT Excluded. Generated reflection outputs are automatically excluded.
Object Filter: Layers	Use effective object layers, then any sorting-layer restrictions.
Include Selected	Only the exact Renderer components in Selected Renderers can pass the layer/sorting filters. An empty selection reflects no selected geometry.
Exclude Selected	Remove the exact selected Renderer components after layer/sorting filtering. Children are not expanded automatically.
Sorting layer selection	Empty includes every sorting layer. Add entries to restrict captured renderer sorting layers. This does not set output draw order.

Example: only environment and player
Assign scenery to Environment, the player to Actors, and shore/UI/decorations to Decoration. Include Environment + Actors; exclude Decoration. Lights can still illuminate selected content without adding their unrelated geometry. Screen-space UI is not reflected.

03 - CAPTURE REGION

09. Set the source rectangle



Example source region: (-10.65, 0, 21.3, 6.8).

Setting	Purpose / action
X	Left edge along the local plane tangent. To centre a 21.3-wide capture, use -21.3 / 2 = -10.65.
Y	Near edge along the source-side normal. 0 starts at the reflection plane. The region stays on the source side; negative Y is sanitized to 0.
W	Source width. The example ends at X + W = 10.65.
H	Source height away from the plane. The example includes 0 to 6.8 source units above a horizontal plane.
Visible capture	The manager also considers the source camera view and surface visibility. Content outside the capture rectangle cannot appear simply by raising opacity.

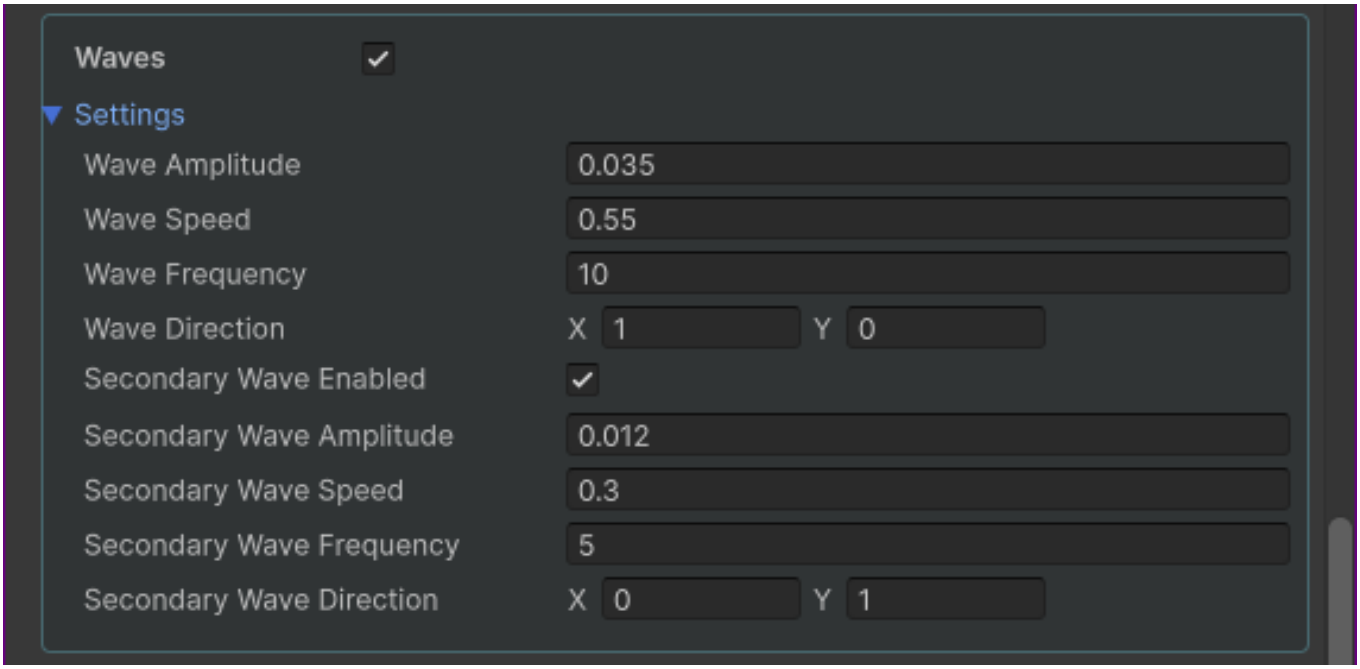
A practical adjustment order

1. Disable optional effects and begin in Planar mode with a visible test sprite.
2. Place the surface at the correct waterline and size its destination using Surface Size or Scene View handles.
3. Set Capture Region to cover the intended source sprites, then inspect the reflection edges and camera framing.
4. Add source margin for wave/offset sampling if distorted edges clip. Keep that margin bounded; a larger capture can require more pixels.

> **For vertical mirrors**
> The fields still describe tangent/normal coordinates, not ordinary local X/Y. The mapping is local (normal, -tangent). Use the geometry diagram on page 7 when framing a vertical mirror.

04 - VISUAL EFFECTS

10. Enable effects and directional waves



A gentle two-wave example. These numbers are artistic settings, not mandatory defaults.

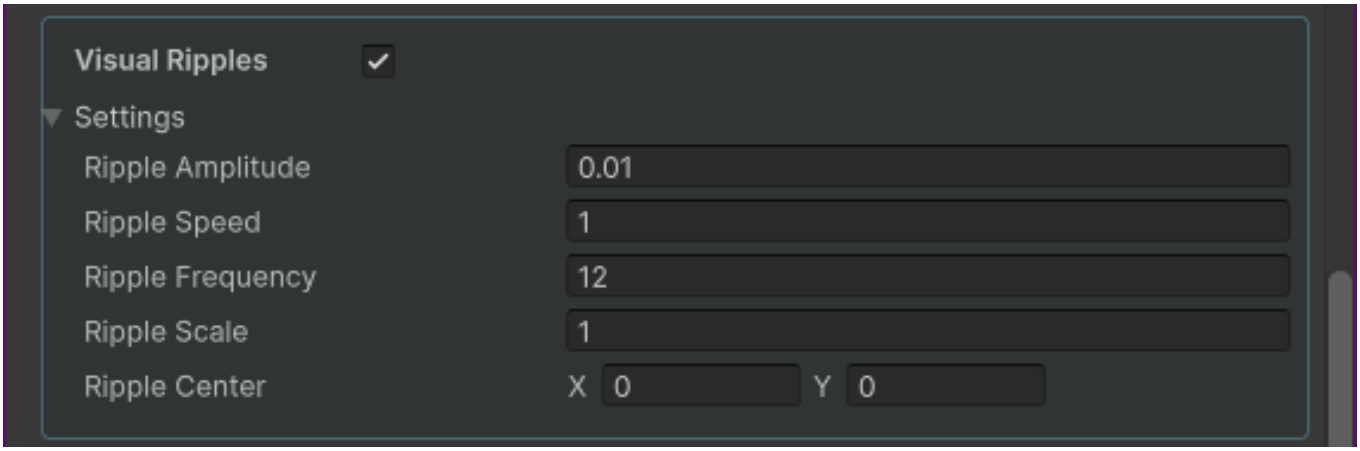
Visual Effects is the master switch. Turning it off bypasses optional shader appearance/distortion/fades while keeping the stored values. Strength and Opacity still control base visibility. Checking a Waves, Ripples, Blur, Fade or Soft Edges card enables the master switch.

Setting	Purpose / action
Waves checkbox	Enables primary directional displacement. Mirror mode suppresses wave and ripple displacement.
Wave Amplitude	Displacement in surface plane units. Begin around 0.015-0.035 for gentle stylized water; too much can reveal capture edges.
Wave Speed	Phase speed in radians per second. A negative speed reverses phase travel; 0 keeps a static pattern.
Wave Frequency	Radians per plane unit. Higher values create more repeated crests across the surface.
Wave Direction	Direction used for phase and displacement; (1,0) and (0,1) separate the two axes. Keep useful nonzero directions.
Secondary Wave	Optional second amplitude/speed/frequency/direction. It has its own enable flag and can run independently of the primary wave.

Visual distortion only
Waves move reflection sampling. They do not animate collision geometry, the original scene or water height. Pixel Safe Distortion quantizes displacement; blur and tight-atlas restrictions are covered later.

04 - VISUAL EFFECTS

11. Configure radial visual ripples



Visual Ripples produces a continuous radial shader pattern.

Setting	Purpose / action
Visual Ripples	Enable the radial pattern and the Visual Effects master switch.
Ripple Amplitude	Radial displacement in plane units. Small values preserve recognizable reflection details.
Ripple Speed	Pattern phase speed in radians per second; negative values reverse phase travel.
Ripple Frequency	Phase repetition with distance from the ripple centre.
Ripple Scale	Scales radial distance before calculating phase. Larger values tighten the pattern.
Ripple Center	Centre in source plane coordinates. Use direct Effects editing plus NotifyVisualEffectsChanged to change it from gameplay.

A gentle starting recipe

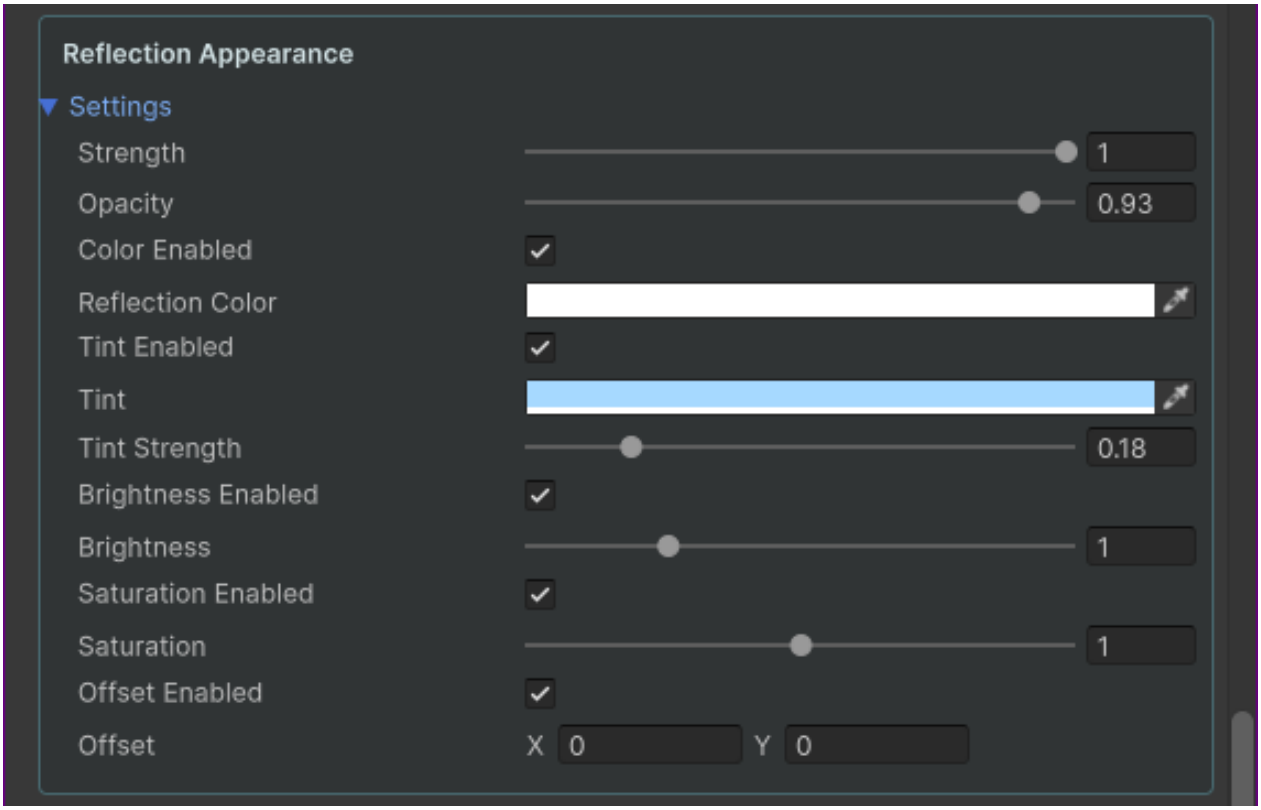
Enable Visual Ripples with amplitude 0.01, speed 1, frequency 12 and scale 1, then move the centre near the intended visual focal point. Combine small primary waves only after the radial effect looks right.

```
// Existing surface reference: WaterReflectionSurface water;
water.SetRipples(true, 0.01f, 12f, 1f, 1f);
water.Settings.Effects.RippleCenter = new Vector2(0f, 1f);
water.NotifyVisualEffectsChanged();
// Or use a built-in ripple configuration:
water.SetRipplePreset(ReflectionRipplePreset.Subtle);
```

Gameplay interaction
This is one continuous visual ripple field per surface. There is no collision-event ripple simulation or automatically tracked list of splash points. Your gameplay script may move the centre or change parameters.

04 - APPEARANCE

12. Colour, tint, brightness and offset



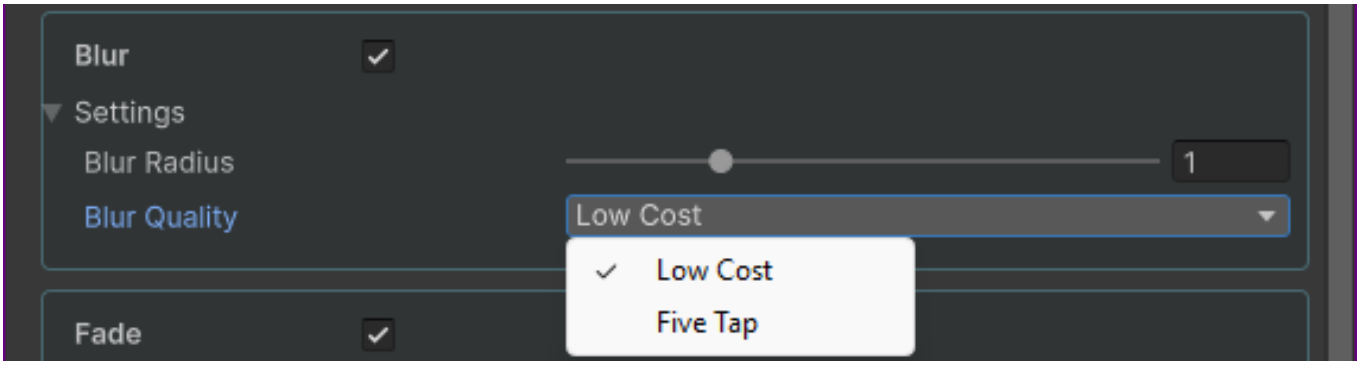
Reflection Appearance controls. Individual appearance effects also need Visual Effects enabled.

Setting	Purpose / action
Strength / Opacity	Both multiply displayed reflection visibility and remain active when Visual Effects is off. Values range from 0 to 1.
Reflection Color	Multiplicative RGBA colour. White preserves the captured colour; colour alpha also affects transparency. Color Enabled controls its use.
Tint / Tint Strength	Multiplicative blend from white towards the chosen RGBA tint, strength 0-1. Tint Enabled retains/disables the configured tint.
Brightness	RGB multiplier, 0-4. 1 preserves brightness; it does not alter source Light 2D settings.
Saturation	0 is grayscale, 1 preserves source saturation, up to 2 increases it.
Offset	Sampling displacement in local plane units. The destination and custom silhouette remain fixed.

A readable reflection
Begin with white colour, modest tint strength, brightness/saturation 1 and zero offset. Lower opacity to blend with a separate water bed. For lighting comparisons, disable optional colour/tint so the native source lighting is easier to evaluate.

04 - BLUR

13. Use blur sparingly



Blur quality choices: Low Cost and Five Tap.

Setting	Purpose / action
Blur checkbox	Enables the optional blur and Visual Effects. Mirror mode can use blur even though it suppresses waves/ripples.
Blur Radius	0-4 capture texels; Lightweight uses source texture texels. It is not a world-space blur distance.
Low Cost	Three-sample approximation in the display shader.
Five Tap	Five-sample approximation for a smoother result at higher sampling cost.
Pixel Art	Pixel Safe Distortion suppresses blur to preserve crisp texels. Use Smooth for an intentionally softened look.
Lightweight atlas	UV distortion/blur for tight-packed sprite geometry is rejected. Use compatible rectangular/unpacked sprites or switch to Planar.

Wet stone floor recipe

1. Use a Planar surface with a shallow destination region over the floor.
2. Apply Wet Floor, or set opacity near 0.45 and a small Low Cost blur radius such as 1.
3. Enable Fade and adjust tint gently so the reflected silhouette remains legible.
4. Compare Quality tiers at your final camera scale. Changing capture resolution also changes the world-space size of a texel blur.

> **This is an approximation**
> The blur is a small fixed tap pattern. It does not create physically accurate rough reflections, multi-pass bloom or a general scene post-processing stack.

04 - FADE

14. Fade the reflection into water



Vertical fade example: start 0.15, end 1 and intensity 0.65.

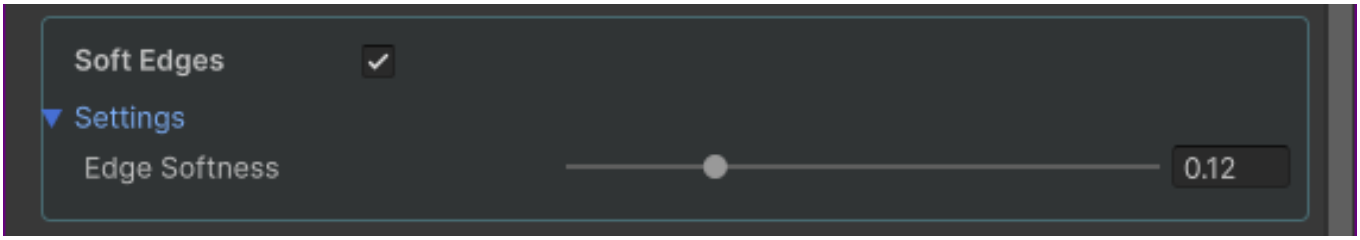
Setting	Purpose / action
Fade checkbox	Enables fading and Visual Effects; at least one axis must be enabled to see axis fading.
Vertical Fade	Normalized depth away from the reflection plane, from 0 at the top edge to 1 at the bottom edge.
Fade Start / End	The normalized range of the vertical transition. End must exceed Start; values are sanitized to keep a valid range.
Horizontal Fade	Symmetric transition from the centre towards either side of the destination.
Horizontal Start / End	0 represents the centre; 1 reaches left/right edges. These are independent of the vertical range in the Inspector.
Fade Intensity	0 means no fading; 1 can fully remove distant/edge output. Lower values retain some visibility at the end.

Vertical and horizontal masks multiply when both are enabled. For a quiet lake, start at 0.15 and end at 1 so near-waterline details remain clear while deep reflection blends out. A puddle often benefits from a horizontal fade as well.

> **API detail**
> SetReflectionFade(vertical, horizontal, start, end, intensity) sets both axis ranges to the same Start/End. To use different horizontal values, edit Effects.HorizontalFadeStart/End and call NotifyVisualEffectsChanged.

04 - SOFT EDGES

15. Blend the destination edges



Soft Edges with Edge Softness 0.12.

Setting	Purpose / action
Soft Edges checkbox	Enables the rectangular-edge blend and Visual Effects.
Edge Softness	Normalized distance inward from the destination rectangle, range 0-0.5. A larger value uses more of the rectangle for the blend.
Custom alpha shape	The shape's own alpha handles its irregular border. Soft Edges still operates on the enclosing destination rectangle; it is not a distance field around every silhouette edge.
Transparent holes	Holes are retained by the sprite alpha mask. Use alpha in the source shape for a soft hole boundary.

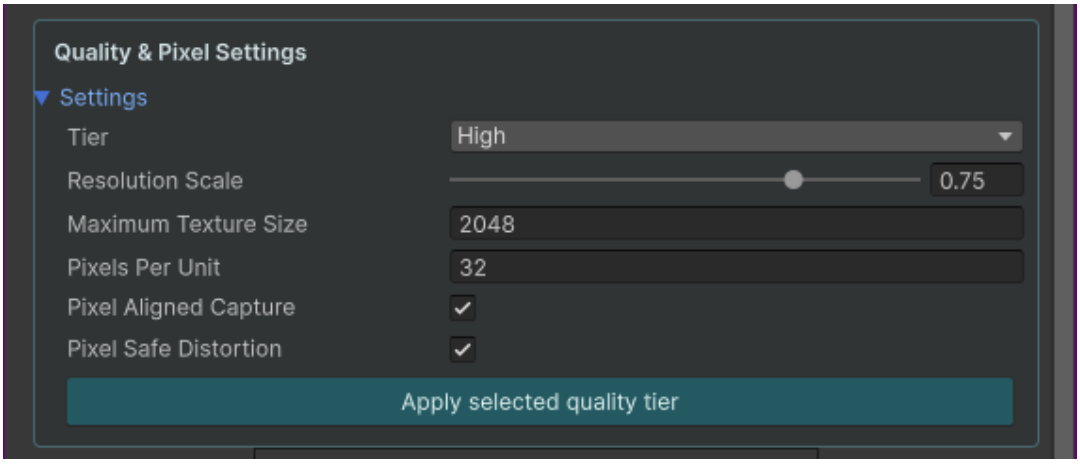
An irregular puddle workflow

1. Prepare an alpha silhouette containing the exact puddle outline and any transparent holes.
2. Assign Surface Shape and click Match sprite aspect. Place the flat plane at the intended top edge of the puddle frame.
3. Exclude ground and shore decoration from captured content, then add an appropriately sorted background under the puddle.
4. Use the sprite alpha for the curved boundary. Add a small fade/blur only if the intended look needs it.

> **Check the alpha import**
> If the reflection stays rectangular, check that the assigned asset is a Sprite, its imported texture retains alpha, and the outside pixels are actually transparent. A white circular picture on an opaque background is still a rectangular alpha mask.

05 - QUALITY

16. Apply a quality tier



Choose a tier, then click Apply selected quality tier.

Setting	Purpose / action
Low	Resolution Scale 0.25; Maximum Texture Size 512; update cap 15/s. Applying Low also disables blur and the secondary wave.
Medium	Resolution Scale 0.5; maximum 1024; update cap 30/s.
High	Resolution Scale 0.75; maximum 2048; update cap 60/s.
Ultra	Resolution Scale 1; maximum 4096; update cap 0 (every eligible frame).
Custom	Use individual settings for your scene. Changing the Tier field alone does not apply the numeric defaults.

Setting	Purpose / action
Resolution Scale	Smooth capture density relative to the camera view; 0.05-1. Pixel Art uses its PPU/grid rules.
Maximum Texture Size	Caps a capture texture axis. Very low caps can reduce sharpness or exact pixel-grid fidelity.
Pixels Per Unit	Pixel Art capture density. Match your source artwork and Pixel Perfect Camera settings.
Pixel Aligned / Safe	Capture snapping and quantized distortion for Pixel Art. These do not repair mismatched source PPU or noninteger scaling.

Quality is a starting point
Compare actual output and frame time. Applying a higher tier after Low does not re-enable blur or the secondary wave.

05 - PIXEL ART

17. Keep pixel artwork crisp

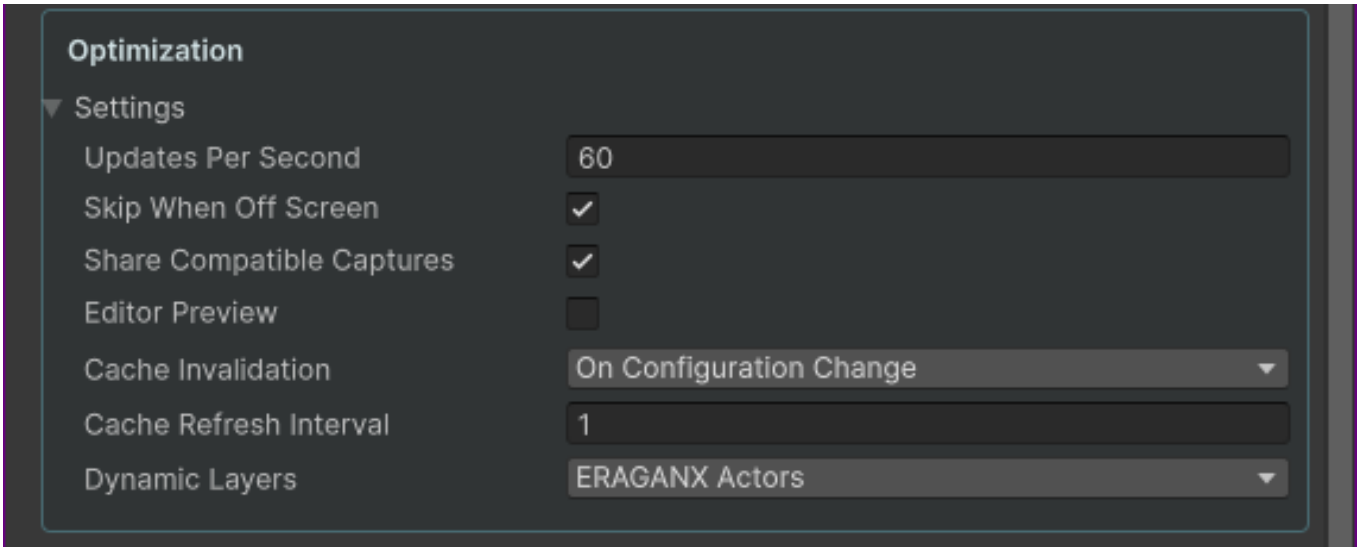
1. Import true pixel artwork as Sprite with Point filtering, no mipmaps and compression that preserves the intended pixels. Use consistent Pixels Per Unit across the art set.
2. Configure your orthographic camera and, when appropriate, URP Pixel Perfect Camera with matching asset PPU and a suitable reference resolution.
3. Choose **Visual Style = Pixel Art** on the surface. Match its Pixels Per Unit to the source. Enable Pixel Aligned Capture and Pixel Safe Distortion.
4. Use positive integer world scaling and a camera/surface position compatible with the PPU grid. Arbitrary rotation or noninteger scale can prevent exact texel correspondence.
5. Keep Maximum Texture Size large enough for the intended capture. Inspect zoom and different resolutions; a cap may force a lower effective density.
6. Start without distortion. Add small waves only after alignment is stable. Pixel Safe Distortion quantizes displacement and suppresses blur.

Setting	Purpose / action
PPU example	32 pixels per unit gives a native grid spacing of 1 / 32 = 0.03125 world units before transform scaling.
Smooth style	Uses Bilinear filtering and continuous effect movement, suitable for painted or high-resolution artwork.
Pixel Art style	Uses Point filtering for captures, grid snapping and optional quantized distortion.
The supplied Pixel demo	Renders the supplied Moonbell illustrations at 320 x 180. It demonstrates low-resolution capture; the illustrations are not hand-authored pixel sprites.

A practical comparison
Compare the original sprite and its reflection at the final output resolution. If only the reflection is soft, check the surface style/PPU/cap. If both are soft, check source imports, camera scaling and the final render resolution.

05 - OPTIMIZATION

18. Control updates and off-screen work



Optimization section. Dynamic Layers is used by Hybrid.

Setting	Purpose / action
Updates Per Second	Positive values cap eligible captures. 0 requests every eligible frame. It does not stop display-shader waves between scene captures.
Skip When Off Screen	Skips capture work when the destination is outside the source camera's relevant view. Scene View preview still uses that camera's frustum.
Share Compatible Captures	Reuses a matching fresh capture conservatively. Geometry/camera/content settings must match; Hybrid and object/sorting-filtered captures are excluded.
Editor Preview	Enables capture outside Play Mode; preview is throttled. Disable it when you only need runtime output.
Cache Invalidation	Manual, On Configuration Change or Interval. Used for stored/static capture content.
Cache Refresh Interval	Seconds between automatic refreshes when Interval is selected.
Dynamic Layers	Hybrid moving content. Only layers inside the effective included mask participate; other included layers form the static cache.

Optimize what is visible
Reduce capture area/resolution, update rate and unnecessary included geometry first. Use caching for content that is genuinely static. Pooling and sharing reduce specific costs; they do not prove zero allocation or a measured mobile speedup.

05 - CACHED AND HYBRID

19. Separate static and moving content

Cached mode

1. Choose Cached for a mostly static scene. The first eligible request captures it; subsequent frames display the stored content.
2. Choose Manual when your gameplay script decides when scene or lighting content changes. Call RefreshCachedReflection after those changes.
3. Choose On Configuration Change when surface configuration changes should refresh the cache. It does not automatically detect every moving renderer or changing source light.
4. Choose Interval for periodic refreshes and set Cache Refresh Interval in seconds. Display effects may continue animating over a frozen image.

Hybrid mode

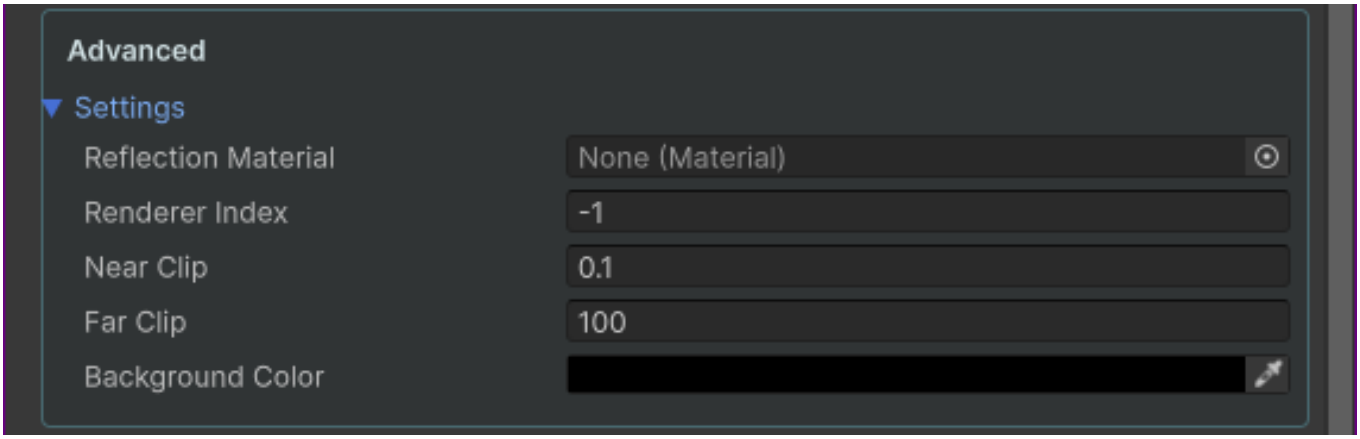
1. Put static scenery and moving actors on distinct object layers. Include both groups and exclude decoration.
2. Select Hybrid and choose the actor layer in Dynamic Layers. Both static and dynamic effective layer groups must be nonempty. Static content is cached; dynamic content follows the update schedule.
3. Refresh the static capture when scenery or its lighting changes. Compare actors against static foreground objects before relying on Hybrid composition.

> **Sorting tradeoff**
> Hybrid composites dynamic over static as two captures. A static transparent foreground object cannot interleave correctly in front of one actor and behind another. Use Planar when that scene ordering, depth or stencil interaction matters.

```
water.SetReflectionMode(ReflectionMode.Hybrid);
water.Settings.Optimization.DynamicLayers =
    LayerMask.GetMask("Actors");
water.NotifySettingsChanged();
// After changing static scenery or its lights:
water.RefreshCachedReflection();
```

05 - ADVANCED

20. Material, renderer and clipping



Advanced defaults: auto material, default renderer and transparent background.

Setting	Purpose / action
Reflection Material	Optional compatible display-material override. None uses the bundled default material for the selected mode. An ordinary water material is not automatically compatible with reflection bindings.
Renderer Index	-1 uses the active URP asset's default renderer. A nonnegative value selects that asset's renderer list entry; it must be a 2D Renderer.
Near Clip	Positive distance along the capture camera depth axis. The source's Z placement must remain inside the capture depth range.
Far Clip	Must be greater than Near Clip. Missing depth-clipped objects can be fixed by adjusting the range and source Z positions.
Background Color	Capture clear colour. Alpha 0 preserves transparent empty regions. Opaque background colour fills those regions inside the silhouette.

Retain the package Resources materials: they hold shader references for runtime use and build inclusion. Reflection output does not modify your source sprite material. For Lightweight, the source material must still meet its limited compatibility rules.

> **Start with defaults**
> Keep material None, renderer -1, near 0.1, far 100 and transparent background unless your scene requires a change. Fix camera, capture region and filtering before writing a custom display shader.

05 - DIAGNOSTICS

21. Read the current status correctly



Configuration accepted means the setup is valid; it is not an automated test result.

Setting	Purpose / action
Disabled	This surface's reflection is turned off.
Awaiting Manager	No active reflection manager is scheduling the surface. Use Connect camera & manager.
Invalid Configuration	A required reference or geometric setting is invalid. Read the message and correct the field.
Unsupported Configuration	The camera, pipeline, transform, renderer or sprite shape/material is outside the supported setup.
Backend Unavailable	The selected rendering path is unavailable for the active pipeline/configuration.
Ready	Current configuration/capture state is accepted. Inspect the visible reflection; Ready does not prove a correct image or device performance.
Capture Failed	The attempted render did not complete. Check Console, compatibility and material/renderer setup.
Refresh cached reflection	Requests fresh scene content, useful for Cached/Hybrid and after static light changes.

1. Read Diagnostics before adding more visual effects. Confirm Source Camera, active manager and effective included layers.
2. If the message says waiting for the next capture, enable Editor Preview or enter Play Mode and keep the destination in the camera view.
3. Use Validate Reflection Setup as a configuration check, then inspect the actual camera output and target build.

06 - PRESETS

22. Apply and save reflection presets

Built-in preset	Starting look
Calm Lake	Gentle primary wave, subtle tint and vertical fade.
Flowing River	Faster primary wave for moving water.
Ocean	Larger primary wave and a secondary wave.
Small Puddle	Soft edges and a slower capture schedule.
Wet Floor	Lower opacity with blur and fade.
Clean Mirror	Mirror mode with full strength/opacity and optional effects off.

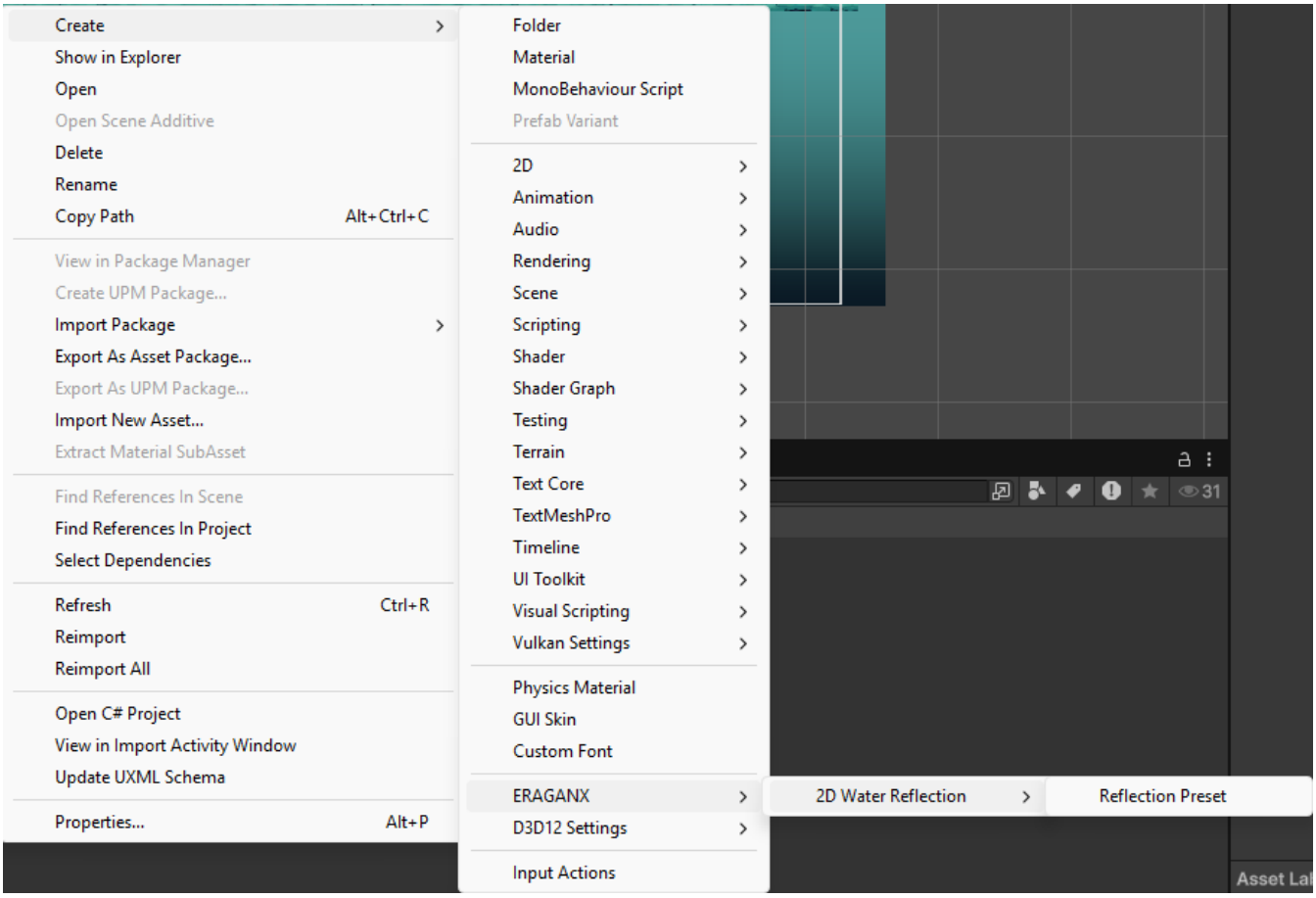
1. Select a Reflection Surface. Drag a built-in or custom Reflection Preset into Apply preset.
2. Inspect the mode and effects it applies. Keep your scene's Source Camera, Surface Size, Surface Shape, Capture Region, selected renderers and layer/sorting filters.
3. Tune the surface for your own scene. Choose **Save custom preset...** and save a new asset in your own project folder.
4. Apply the custom preset to another surface or call ApplyReflectionPreset at runtime. Presets copy settings so surfaces do not share one mutable Effects object.

> **What a preset preserves**
> Camera, enabled state, position/size/shape, capture rectangle, renderer selections, object/sorting layers, output sorting and Advanced capture bindings remain scene-specific. Hybrid Dynamic Layers and Editor Preview are retained. Mode, orientation, style, surface category, effects, quality and reusable optimization settings are copied.

Changing Surface Type alone does not apply the corresponding preset. A Clean Mirror preset uses its saved orientation; inspect the orientation if you want a vertical mirror. Keep user presets outside the bundled Presets folder when updating the package.

06 - PRESETS

23. Create a Reflection Preset asset



Project window > Create > ERAGANX > 2D Water Reflection > Reflection Preset.

1. Create a folder such as Assets/Game/ReflectionPresets and right-click it in the Project window.
2. Choose the menu shown above. Give the asset a useful name such as Quiet Blue Lake.
3. Configure Mode, Visual Style, Surface Type and Mirror Orientation. Expand Effects, Quality and Optimization to set its reusable values.
4. Ensure Visual Effects Enabled is on when your preset needs tint, waves, blur or fade. Values on an inactive effect are retained but not applied.
5. Apply the preset to a test surface, verify its actual image, and save the asset. Use Save custom preset on a configured surface when you prefer to capture its existing values.

ScriptableObject is a settings asset
A preset is not a scene or water texture. It does not contain Source Camera, scene Renderer selections or surface geometry. The reusable asset can be referenced by your gameplay scripts.

06 - PRESET INSPECTOR

24. Understand the preset Effects group



Preset Effects overview. Individual controls are explained on pages 14-19.

Same rendering controls

The SO stores the surface's effect values: strength, opacity, colour/tint, brightness, saturation, offset, both waves, ripples, blur, fades and edge softness.

Master effect flag

Visual Effects Enabled must be checked to use optional effect settings. In this screenshot it is unchecked, so stored tint/wave/blur values do not all become active simply because they are visible.

Individual flags

Enabled fields let you keep an effect configuration without applying it. For example, Tint Strength 0 preserves the original captured colour even if Tint Enabled is checked.

Mode rules still apply

Mirror suppresses waves/ripples. Pixel Art with Pixel Safe Distortion suppresses blur. Lightweight content must meet its source-sprite compatibility rules.

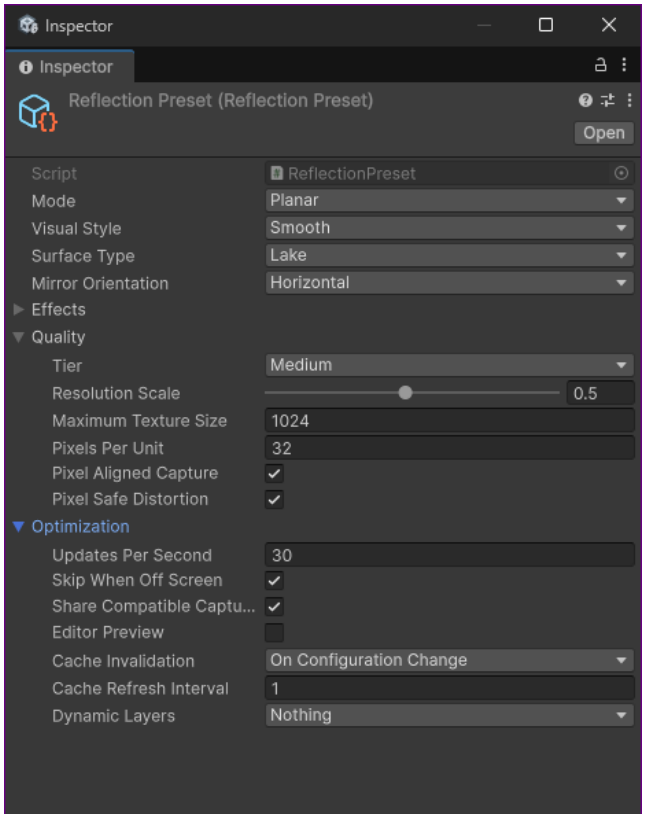
Applying the preset

Copies effect settings into the chosen surface. Scene bindings are preserved; there is no shared mutable effect state between applied surfaces.

Use the detailed effect pages for units and ranges; this overview shows how the same fields are stored in a preset.

06 - PRESET INSPECTOR

25. Preset quality and optimization



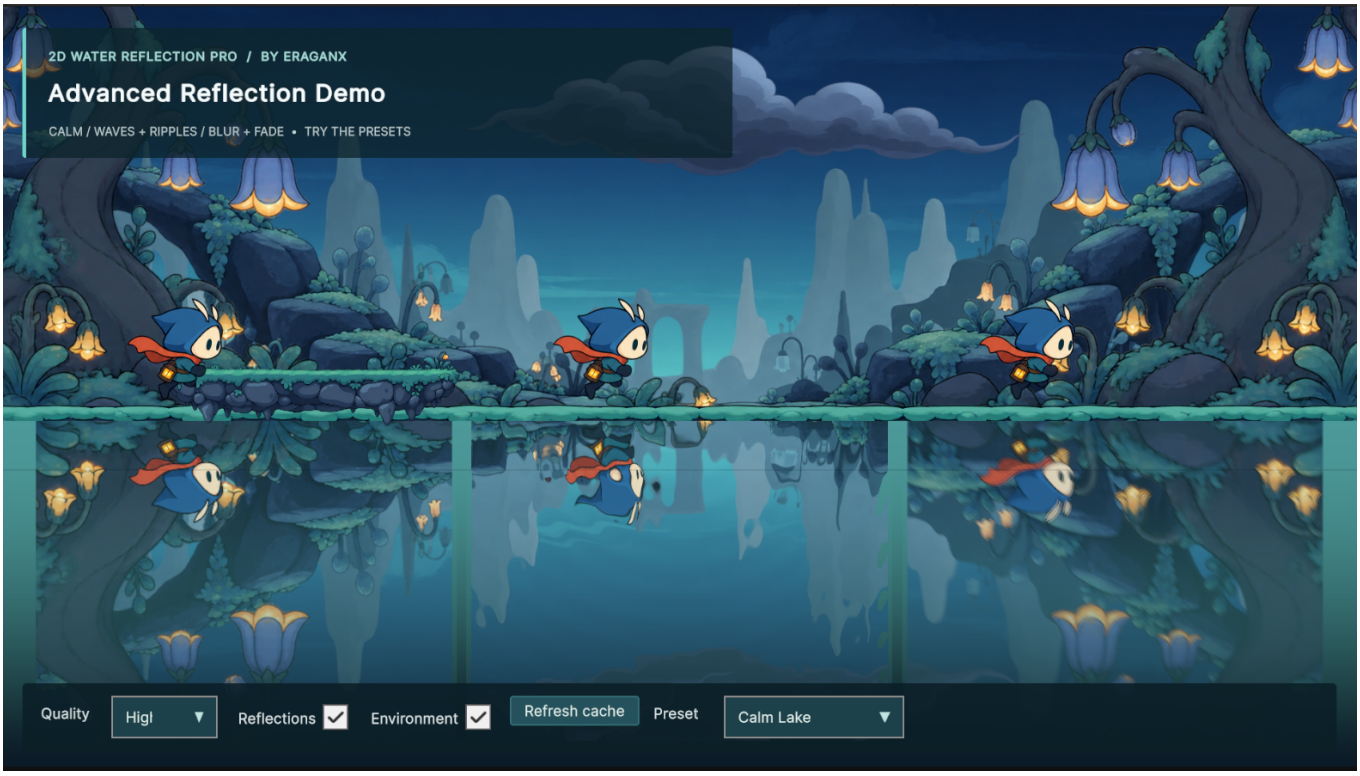
Preset Quality and Optimization groups. Dynamic Layers is scene-specific.

Setting	Purpose / action
Quality values	Tier, Resolution Scale, Maximum Texture Size, Pixels Per Unit and pixel flags are stored as numeric settings.
Tier field	On a preset, choosing a tier label does not automatically replace its numeric values. Configure matching values or capture a surface after applying the desired tier.
Optimization values	Update rate, off-screen skipping, capture sharing, cache policy and interval are copied when applied.
Editor Preview	The receiving surface keeps its existing Editor Preview value. A preset does not turn runtime-only surfaces into continuous Editor captures.
Dynamic Layers	The receiving Hybrid surface keeps its own Dynamic Layers mask. CaptureFrom deliberately clears the preset's stored mask because layer indices vary by project.

A custom performance preset
Apply Low/Medium/High to a surface first, inspect its effects/update rate, then Save custom preset. Reapply to another surface and confirm its retained Hybrid layer selection and camera bindings.

07 - SAMPLES

26. Explore the actual sample output



Supplied actual Game view: Advanced Reflection Demo, with three independent surfaces and UI.

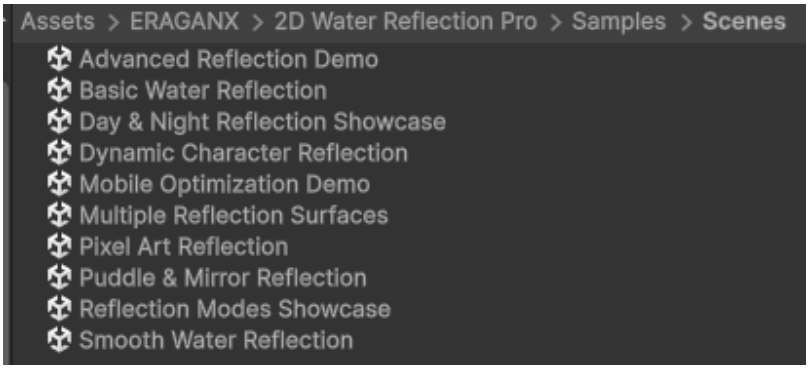
This is the rendered sample scene, rather than the cover illustration. The Moonbell scenery and animated Niko sprites appear in the source scene and in water. The three panels demonstrate calm water, waves/ripples, and blur/fade/tint.

Setting	Purpose / action
Quality	Calls the package quality API for the listed demo surfaces.
Reflections	Toggles the listed reflection outputs.
Environment	Excludes/includes the environment layer in reflections. It does not hide the original scene scenery.
Refresh cache	Requests fresh stored captures. In the Modes sample, this unfreezes the Cached panel until its next stored capture.
Preset	Available in Advanced; applies a chosen appearance/performance preset to its controlled surfaces.

> **Use your own content**
> The included illustrations demonstrate the tool. Replace source sprites/materials and layer selections for your game. SampleCameraFraming preserves a 16:9 demo composition; remove/disable that sample helper when authoring different camera framing.

07 - SAMPLES

27. The ten official scenes



Package folder: Samples/Scenes.

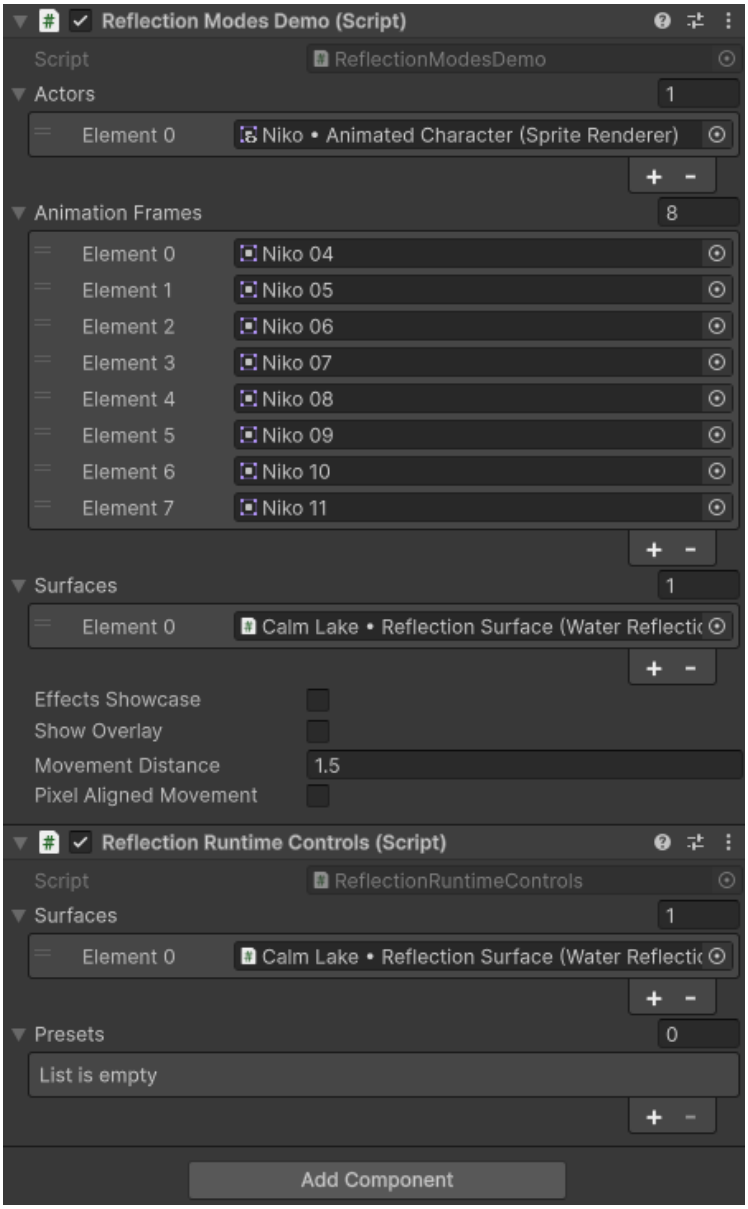
Scene	Demonstration
Basic Water Reflection	Layered Endless Jungle, calm water and animated Niko.
Pixel Art Reflection	Moonbell illustrations at 320 x 180 with Point capture and grid movement.
Smooth Water Reflection	Moonbell grove with Bilinear capture and two gentle waves.
Dynamic Character Reflection	Eight-frame animated Niko beside undistorted jungle water.
Multiple Reflection Surfaces	Three independent capture regions at different update rates.
Puddle & Mirror Reflection	Soft puddle, wet floor and a framed vertical clean mirror.
Mobile Optimization Demo	Low-quality settings and moving sprites. It is not a measured device benchmark.
Advanced Reflection Demo	Calm / waves-ripples / blur-fade-tint, with preset controls.
Day & Night Reflection Showcase	Actual Global/Point Light 2D changes, phase and local-light controls.
Reflection Modes Showcase	Planar, Lightweight, Cached, Hybrid and Mirror side by side.

Hierarchy groups: **01 Cameras & Systems**, **02 Environment**, **03 Characters**, **04 Reflection Surfaces**, optional **05 Lighting**, and **06 Demo Controls & UI**. Environment includes Background, Scenery and Shore & Foreground. Water Beds & Frames groups the reflection decoration.

> **Start with a copy**
> Open and duplicate a sample before customizing it. The sample rebuild command restores authored bundled content. No redundant separate Effects Showcase is needed; Advanced demonstrates those effects.

07 - DEMO SCRIPTS

28. Animation and runtime controls



Sample Controls object: animation demo and runtime control references.

Actors
The SpriteRenderers animated by ReflectionModesDemo. Drag the actual character renderers here.

Animation Frames
The sample uses Niko 04-11 as eight running frames. The component swaps sprites and demonstrates movement.

Surfaces
Reflection surfaces targeted by the animation showcase and runtime control component. Keep these scene references assigned.

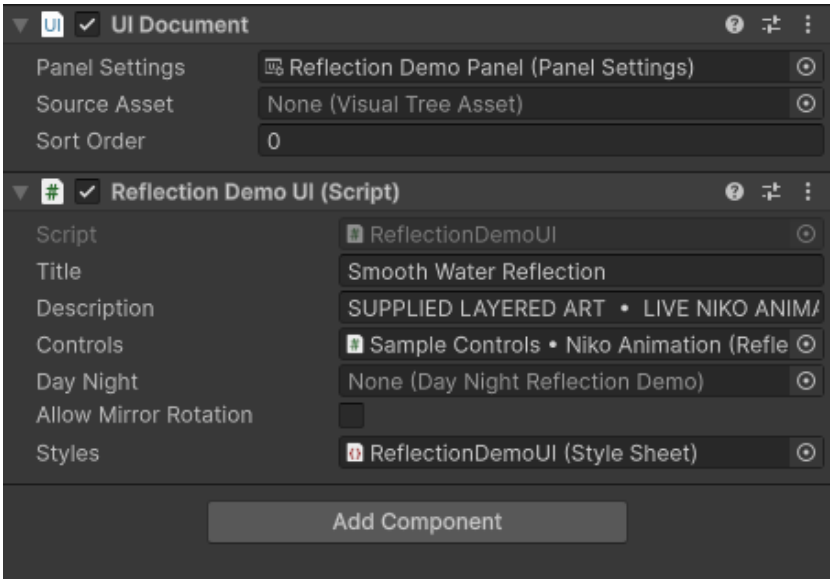
Movement
Movement Distance and Pixel Aligned Movement tune demo motion. They are sample helpers, not a platformer controller.

Runtime Controls
ReflectionRuntimeControls exposes public methods for quality, visibility, environment exclusion, presets and cache refresh.

> **An empty preset list can be intentional**
> The screenshot shows no presets on the Smooth demo controls. Advanced supplies preset references. The core reflection still works when a demo-only preset selector is absent.

07 - DEMO UI

29. UI Toolkit and Day/Night controls



UI Document plus Reflection Demo UI. Source Asset None is intentional: the UI is built in code.

Setting	Purpose / action
UI Document	Uses Reflection Demo Panel. A separate UXML source is not required for this code-built demo interface.
Title / Description	Demo heading and explanatory text. Change them for your own sample copy.
Controls	Reference to ReflectionRuntimeControls. This connects the UI actions to the listed surfaces.
Styles	The bundled ReflectionDemoUI.uss style sheet. Panel settings also use the bundled theme.
Day Night	Assign the DayNightReflectionDemo in the lighting showcase. Other scenes can leave this field None.
Allow Mirror Rotation	Exposes mirror orientation in the Modes sample. The framed mirror sample keeps its fixed orientation so the frame remains aligned.

Day & Night adds phase selection, pause/resume, cycle duration and local-light toggles. Lit scene sprites respond to actual Global and Point Light 2D output; sky/cloud materials may be unlit. Lantern glow painted into an illustration remains part of the artwork.

> **Lighting scope**
> Planar/native captures reproduce compatible 2D scene lighting. Lightweight does not reproduce lit materials. Cached/static Hybrid lighting needs a refresh when lights change. Complex shadows, normal maps, SpriteMasks and blend-style combinations need additional validation.

08 - RUNTIME API

30. Connect your gameplay script

1. Create a C# file whose filename matches the class below. Add an ERAGANX.WaterReflection2D reference to your own asmdef when using one.
2. Attach the script to an empty GameObject. Assign the existing Reflection Surface, Source Camera and optional preset in the Inspector.
3. Keep one active WaterReflectionManager in the scene. The surface reference lets your gameplay/UI code call public methods.

```
using UnityEngine;
using ERAGANX.WaterReflection2D;

public sealed class GuideWaterControls : MonoBehaviour
{
    [SerializeField] WaterReflectionSurface water = null;
    [SerializeField] Camera sourceCamera = null;
    [SerializeField] ReflectionPreset calmPreset = null;

    void Start()
    {
        if (!water || !sourceCamera) return;
        water.SetSourceCamera(sourceCamera);
        water.SetSurfaceSize(new Vector2(10f, 3f));
        water.Settings.Capture.Region =
            new Rect(-5f, 0f, 10f, 3f);
        water.NotifySettingsChanged();
        if (calmPreset)
            water.ApplyReflectionPreset(calmPreset);
        water.SetReflectionEnabled(true);
    }

    public void UseLowQuality()
    {
        if (water)
            water.SetReflectionQuality(ReflectionQuality.Low);
    }

    public void Refresh()
    {
        if (water) water.RefreshCachedReflection();
    }
}
```

Calling methods
The public button methods can be connected to UI events or called by another gameplay component. No API key, network service or package singleton is required. Keep references to the particular surface you want to control.

08 - RUNTIME API

31. Change visual effects at runtime

The effect setters sanitize values and enable Visual Effects where needed. Strength/Opacity do not force optional effects on. These calls affect displayed reflection appearance rather than the original scene or water physics.

```
using UnityEngine;
using ERAGANX.WaterReflection2D;

public sealed class GuideWaterEffects : MonoBehaviour
{
    [SerializeField] WaterReflectionSurface water = null;

    public void UseGentleWater()
    {
        if (!water) return;
        water.SetReflectionMode(ReflectionMode.Planar);
        water.SetReflectionStrength(1f);
        water.SetReflectionOpacity(0.85f);
        water.SetWaves(true, 0.025f, 0.55f, 10f,
                       Vector2.right);
        water.SetSecondaryWave(true, 0.01f, 0.3f, 5f,
                               Vector2.up);
        water.SetRipples(false, 0.01f, 12f, 1f, 1f);
        water.SetReflectionTint(
            new Color(0.65f, 0.85f, 1f, 1f), 0.18f);
        water.SetReflectionBrightness(1f);
        water.SetReflectionSaturation(1f);
        water.SetReflectionFade(true, false,
                                0.15f, 1f, 0.65f);
        water.SetReflectionBlur(false, 1f);
    }

    public void UseCleanMirror()
    {
        if (!water) return;
        water.SetReflectionMode(ReflectionMode.Mirror);
        water.SetMirrorOrientation(
            ReflectionMirrorOrientation.Vertical);
        water.SetReflectionStrength(1f);
        water.SetReflectionOpacity(1f);
        water.SetVisualEffectsEnabled(false);
    }
}
```

Setter parameter order
SetWaves takes amplitude, speed, frequency, direction after enabled. SetRipples takes amplitude, frequency, speed, scale after enabled. These signatures intentionally differ; use the quick reference when wiring calls.

08 - RUNTIME API

32. Filter content and set a shape

Use masks selected in your script Inspector instead of hardcoded demo layer numbers. Object layers and sorting-layer IDs are different. Direct capture/optimization edits need NotifySettingsChanged after the batch.

```
using UnityEngine;
using ERAGANX.WaterReflection2D;

public sealed class GuideWaterSelection : MonoBehaviour
{
    [SerializeField] WaterReflectionSurface water = null;
    [SerializeField] LayerMask included = 0;
    [SerializeField] LayerMask excluded = 0;
    [SerializeField] LayerMask movingActors = 0;
    [SerializeField] Sprite puddleShape = null;
    [SerializeField] Renderer[] selectedRenderers = null;

    public void UseHybridLayers()
    {
        if (!water) return;
        var settings = water.Settings;
        settings.Capture.IncludedLayers = included;
        settings.Capture.ExcludedLayers = excluded;
        settings.Capture.ObjectFilter =
            ReflectionObjectFilter.Layers;
        settings.Optimization.DynamicLayers = movingActors;
        water.SetReflectionMode(ReflectionMode.Hybrid);
        water.NotifySettingsChanged();
        water.SetSurfaceShape(puddleShape, 0.001f);
        water.RefreshCachedReflection();
    }

    public void UseSelectedSprites()
    {
        if (!water) return;
        water.Settings.Capture.ObjectFilter =
            ReflectionObjectFilter.IncludeSelected;
        water.Settings.Capture.SelectedRenderers =
            selectedRenderers;
        water.SetReflectionMode(ReflectionMode.LightweightSprite);
        water.NotifySettingsChanged();
    }
}
```

Selection remains restrictive
Selected renderers must also pass included/excluded and sorting-layer filters. Lightweight additionally requires compatible source sprites. SetSurfaceShape(null) restores a rectangle; assigning a shape does not automatically match its aspect ratio.

08 - RUNTIME API

33. Events, cache refresh and direct edits

```
using UnityEngine;
using ERAGANX.WaterReflection2D;

public sealed class GuideWaterEvents : MonoBehaviour
{
    [SerializeField] WaterReflectionSurface water = null;

    void OnEnable()
    {
        if (!water) return;
        water.DiagnosticChanged += OnDiagnostic;
        water.ReflectionUpdated += OnReflectionUpdated;
    }
    void OnDisable()
    {
        if (!water) return;
        water.DiagnosticChanged -= OnDiagnostic;
        water.ReflectionUpdated -= OnReflectionUpdated;
    }
    void OnDiagnostic(ReflectionDiagnostic info)
    {
        Debug.Log(info.Status + ": " + info.Message, this);
    }
    void OnReflectionUpdated(WaterReflectionSurface source)
    {
        // Optional UI/status work. Do not start nested renders.
    }
    public void StaticSceneryChanged()
    {
        if (water) water.RefreshCachedReflection();
    }
}
```

Setting	Purpose / action
Capture/quality edits	Batch direct Settings edits, then NotifySettingsChanged. Structural changes invalidate capture resources/state as needed.
Effects-only edits	Change Settings.Effects, then NotifyVisualEffectsChanged. This sanitizes display effects without refreshing cached scene content.
Shape edits	Use SetSurfaceShape. The shape is a display mask and can reuse the existing capture. SetSurfaceSize separately if you want a new aspect.

Lifecycle
Unsubscribe events when your listener is disabled. ReflectionUpdated reports a completed displayed capture update, not proof of a visually correct result. Avoid changing scene renderers/lights or making unrelated nested render calls inside capture callbacks.

09 - API REFERENCE

Surface, mode and capture API

Public member	Use
SetReflectionEnabled(bool)	Toggles this surface without deleting its configuration.
SetSourceCamera(Camera)	Assigns the source camera and updates capture configuration.
SetSurfaceSize(Vector2)	Changes destination plane width/height.
SetSurfaceShape(Sprite, float = 0.001f)	Sets alpha silhouette and cutoff. null restores a rectangle. Does not resize automatically.
SetOutputSorting(int, int)	Output sorting-layer ID and order. Lightweight preserves source sorting.
SetReflectionMode(ReflectionMode)	Planar / LightweightSprite / Cached / Hybrid / Mirror.
SetMirrorOrientation(enum)	ReflectionMirrorOrientation.Horizontal or Vertical. Vertical is a Mirror orientation.
SetVisualStyle(enum, int = 32)	ReflectionVisualStyle.Smooth or PixelArt, plus Pixels Per Unit.
SetReflectionQuality(enum)	Applies ReflectionQuality Low/Medium/High/Ultra numeric defaults. Custom keeps individual values.
SetReflectionUpdateRate(float)	Caps eligible captures per second; 0 means every eligible frame.
ApplyReflectionPreset(preset)	Copies reusable preset groups while retaining scene bindings.
RefreshCachedReflection() / RequestRefresh()	Requests fresh cached/static or live capture content.
NotifySettingsChanged()	Sanitizes and informs the renderer after direct structural settings changes.
TryValidateConfiguration(out diagnostic)	Returns configuration support and a ReflectionDiagnostic. It does not inspect rendered pixels.
Diagnostic / events	Read Diagnostic. Subscribe to DiagnosticChanged(Action<ReflectionDiagnostic>) and ReflectionUpdated(Action<WaterReflectionSurface>).

Namespace: ERAGANX.WaterReflection2D. **Runtime assembly:** ERAGANX.WaterReflection2D. Use your actual scene references rather than finding or creating a manager every frame.

09 - API REFERENCE

Effect methods and editable groups

Public method	Arguments / action
SetVisualEffectsEnabled(bool)	Master optional shader effects.
SetReflectionStrength(float) / SetReflectionOpacity(float)	Base visibility multipliers, each 0-1.
SetReflectionColor(Color)	RGBA multiplier; enables Color and Visual Effects.
SetReflectionTint(Color, float)	Tint colour and 0-1 strength; enables tint/effects.
SetReflectionBrightness(float) / SetReflectionSaturation(float)	Brightness 0-4; saturation 0-2.
SetReflectionOffset(Vector2)	Local-plane sampling displacement.
SetWaves(bool, float, float, float, Vector2)	enabled, amplitude, speed, frequency, direction.
SetSecondaryWave(bool, float, float, float, Vector2)	Same order as primary waves; independent enable.
SetWaveSpeed(float) / SetWaveDistortion(float)	Primary speed or amplitude. SetWaveDistortion also updates WavesEnabled.
SetRipples(bool, float, float, float, float)	enabled, amplitude, frequency, speed, scale.
SetRipplePreset(enum)	ReflectionRipplePreset.Subtle or Strong.
SetReflectionBlur(bool, float, enum = LowCost)	enabled, radius in texels, ReflectionBlurQuality.
SetReflectionFade(bool, bool, float, float, float = 1)	vertical, horizontal, start, end, intensity; copies range to both axes.
SetReflectionEdgeSoftness(bool, float)	Rectangular-edge blend enable and normalized softness.
NotifyVisualEffectsChanged()	Sanitizes direct Effects edits without invalidating the scene capture.

Settings groups: Capture (Region, IncludedLayers, ExcludedLayers, ObjectFilter, SelectedRenderers, SortingLayerIds, RendererIndex, NearClip, FarClip, BackgroundColor); Effects; Quality; Optimization. Directly set a field only with the appropriate notification afterward.

10 - TROUBLESHOOTING

Fix setup, selection and geometry

Symptom	Check / fix
Tools menu is missing	Check Console compile errors, Editor assembly import and URP dependencies. The menu appears after package scripts compile.
No reflection in Edit Mode	Enable Editor Preview or enter Play Mode. Assign Source Camera and ensure one active manager. Keep the surface inside the source-camera view.
No reflection in Play Mode	Check active/enabled surface and manager, supported camera/pipeline, nonzero Strength/Opacity and Diagnostics.
Everything is transparent	Check Included/Excluded layers, an empty Include Selected list, sorting restrictions, source Region, depth clip planes and sprite alpha.
Unwanted scenery reflects	Exclude its object layer or select exact renderers. Decoration on an included layer will still pass until filtered.
Mask says Mixed...	A multi-layer mask can display Mixed... normally. Expand the mask to inspect checked layers. Multi-object editing can also have mixed values.
Output hidden behind a sprite	Change Output sorting layer/order or water-bed ordering. Source inclusion does not control destination draw priority.
Reflection appears on wrong side	Horizontal extends below local Y = 0. A vertical Mirror extends left of local X = 0. Check mode/orientation and positive transform scale.
Sprite is cut off	Expand or reposition Capture Region; check source camera view, Near/Far Clip and destination sizing. Reduce large wave/offset values while diagnosing.
Shape stays rectangular	Assign an alpha Sprite and retain texture alpha. None means rectangle; an opaque-background image also has rectangular alpha.
Circle becomes an ellipse	Click Match sprite aspect or set destination height = width x original sprite height / width.
Tight mask warning	Use an unpacked or rectangular-packed shape Sprite. Tight Packing is intentionally unsupported for the shape mapping.

Isolate the cause
Use Planar with Visual Effects off, one source sprite and a transparent background. Correct camera, capture geometry and filtering before reintroducing caches, masks and appearance effects.

10 - TROUBLESHOOTING

Fix appearance, caching and performance

Symptom	Check / fix
Waves/tint values do nothing	Enable Visual Effects and the individual feature. A zero amplitude/tint strength gives no effect. Mirror intentionally suppresses waves/ripples.
Blur is missing in Pixel Art	Pixel Safe Distortion suppresses blur. Use Smooth or choose an intentional non-pixel-safe setup if softness is required.
Cached actor or light is frozen	That content is stored. Refresh manually, use Interval, place actors in Hybrid Dynamic Layers, or use Planar.
Hybrid actor draws over foreground	Dynamic composites above static. Use Planar for interleaved transparent/static foreground ordering.
Lightweight rejects an object	Use Simple Sprite-Unlit-Default, no SpriteMask/SpriteSkin/property block/split-alpha texture. Use Planar for lit, Tilemap, particle, Sliced/Tiled or custom-rendered content.
Lightweight effects reject atlas	Tight-packed distortion/blur is unsupported. Use compatible rectangular/unpacked geometry or Planar.
Reflection is soft or pixel grid swims	Check source PPU/filtering, surface style/PPU, camera grid, integer scale and texture cap. Compare at final output resolution.
Quality label changed only	Click Apply selected quality tier, or call SetReflectionQuality. Presets store numeric values independently of the label.
Low removed blur/secondary wave	Applying Low intentionally disables both. Re-enable the wanted effect explicitly after switching to a higher tier.
Samples use wrong layers after import	Install Demo Layers adds names only. Rebuild Official Samples remaps bundled objects/masks to available project slots. Preserve customized copies first.
UI Document Source Asset is None	The demo UI is built in code. Check Panel Settings, Controls reference and Styles rather than assigning an arbitrary UXML.
Capture work is expensive	Reduce region, resolution, update rate and included geometry. Use off-screen skipping and static caches where appropriate; profile the actual target device.

11 - COMPLETION

Your first-scene checklist

1. Confirm Unity/URP dependencies and a supported orthographic Base +Z camera. Resolve package-origin Console errors before inspecting the image.
2. Create one Reflection Surface and an active manager. Put its Transform at the intended waterline; keep positive orthogonal scale.
3. Set Source Camera, destination Surface Size and source Capture Region. Verify a test sprite in Planar with Visual Effects off.
4. Choose included/excluded object layers and any exact renderer/sorting filters. Check that decoration and other water outputs do not appear.
5. Assign an optional alpha shape, match aspect and check transparent holes/edges. Confirm the plane stays in the expected position.
6. Choose a preset or enable effects individually. Compare opacity/tint, waves/ripples, blur/fade and edge treatment at the final camera scale.
7. Apply and inspect quality values. Decide update rate/cache mode using the scene's actual moving content and transparency ordering.
8. Wire runtime methods using explicit surface references. Subscribe/unsubscribe events correctly. Refresh static caches after scenery or lighting changes.
9. Save Edit Mode scene/prefab/preset changes. Test disable/re-enable, mode switching, camera movement, sample layers and resizing.
10. Build and test on your target device/API, inspect actual camera/UI output and profile frame time/resource use. A Windows Editor check is not a device result.

Keep the integration focused
The core tool handles reflection rendering. Your game owns movement, physics, camera follow, splash events, audio, save data and gameplay UI. Keep custom content in your own folders so package updates remain straightforward.

11 - FAQ AND GLOSSARY

Common questions and useful terms

Question	Answer
Only side-scrollers?	No. Compatible orthographic 2D scene composition is the requirement. Adventure backgrounds, menus, decorative water and mirrors can use it.
Only rectangular water?	No. A Sprite alpha silhouette supports circles, triangles, irregular outlines, holes and soft boundaries. The underlying reflection plane remains flat.
Can I exclude layers?	Yes. Included minus Excluded, exact Renderer selection and sorting-layer restrictions control captured content.
Is this a texture pack?	It is a reflection rendering tool. Sample artwork helps demonstrate it; you may replace that artwork.
Does it include physics?	No buoyancy, fluid simulation, collision waves or automatic splash-event ripples.
Can lights reflect?	Compatible native 2D lighting appears in scene captures. Lightweight is an unlit sprite path; cached lighting must be refreshed.
Does a preset replace my scene?	No. It copies reusable visual/performance groups and keeps scene geometry and bindings.

Term	Meaning
Plane / destination	Flat mirror line / displayed reflection frame on the opposite side.
Capture / cache	Rendered source image / retained image reused until refreshed.
Object layer / sorting layer	Capture membership mask / renderer draw-order category.
PPU / texel	Pixels per world unit / one pixel of a sampled texture.
Alpha / preset	Transparency used for silhouette / reusable ScriptableObject settings.

12 - SOURCE AND SCOPE

Version scope, sources and support

This guide was prepared in English on **11 October 2026** from the current ERAGANX package source, 24 supplied Unity screenshots and three supplied branding images. Field names and C# signatures match the inspected source. Screenshots show example values and may include tooltips or intentional empty fields.

Source	Scope
Package root	Assets/ERAGANX/2D Water Reflection Pro/
Runtime source	Runtime/Surfaces, Core, API, Effects, Rendering, ReflectionModes, Optimization and Presets.
Editor source	Editor/Wizards/ReflectionSetup.cs; Editor/Inspectors; Editor/PresetEditors; Editor/SceneTools and sample authoring tools.
Samples	Samples/Scenes, prefabs, runtime demo scripts, Materials and publisher-supplied Sprites/Realistic artwork.
Inspected versions	Unity 6000.3.13f1; URP 17.3.0; Windows Editor with D3D11.

Compatibility limits to keep in view

Perspective/tilted cameras, XR, camera stacks, screen-space UI reflection, arbitrary 3D mirrors and nonorthogonal/negative surface transforms are unsupported. Hybrid cannot reproduce interleaved static/dynamic transparency. Lightweight supports a deliberately limited unlit sprite path. Shape alpha changes silhouette rather than curved mirror geometry.

Global/Point Light 2D and scoped Windows Editor rendering checks have been exercised. Player/mobile/device/API performance, long resource soaks, complex shadows/normal maps/masks/blend styles and rotated atlas integration require further validation. This guide does not claim those unrun checks passed.

Official Unity references

Unity 6.3: 2D lighting in URP
Unity 6.3: Sprite texture import settings
Unity 6.3: Pixel Perfect Camera reference

> **Getting help**
> Contact ERAGANX through the support details on the product's Asset Store publisher/product page. Provide Unity and URP versions, target platform/API, reflection mode, Diagnostics message and a small reproducible scene. Use the listing for the latest support contact information.