

2D

ERAGANX

2D PARALLAX PRO

Dynamic 2D Environments Made Easy



ERAGANX · UNITY PACKAGE GUIDE

The complete guide from the beginning

Scene setup, every Control Panel setting, and
C# API usage - step by step.

ENGLISH EDITION

SCREENSHOT GUIDE

C# EXAMPLES

START HERE

How to use this guide



This guide is based on the current 2D Parallax Pro package source and the 18 supplied images. Field names match the Unity interface so you can find them easily. Complete the Editor setup first, then use the C# examples you need.

Foundation and installation	3 - 5
Environment, camera, layers, and looping	6 - 11
Clock and environment colours	12 - 14
Lighting, URP, Sun/Moon, and rays	15 - 19
Wind, fog, clouds, and particles	20 - 24
Quality and preset creation	25 - 26
Demo controls and sample scenes	27 - 28
C# API setup and full examples	29 - 38
API quick reference	39 - 41
Troubleshooting	42 - 43
Guided checklist and glossary	44
Sources and version scope	45

Suggested reading path

If you are new to the package, complete the first scene setup and basic parallax exercise. If you already have a scene, go to Layers, motion modes, and the API sections. Screenshot values are examples; adjust them for your artwork and gameplay.

Package source release: 1.0.0-development. This guide does not change the package code. API examples are also supplied as separate scripts.

01 · FOUNDATION

How does parallax work?

As the camera moves, background layers move at different rates to create a sense of depth. In this package, **Factor** describes how far a layer moves in world space relative to camera displacement. Choosing Background or Foreground does not change the factor.

```
layerPosition = baseline + (camera - reference) * factor + offset + drift
```

Factor X	When the camera moves +10 units	Apparent movement on screen
0	Layer +0 units	-10 units relative to the camera; fixed in world space.
0.5	Layer +5 units	Moves -5 units on screen.
0.8	Layer +8 units	Moves -2 units on screen; resembles a distant background.
1	Layer +10 units	Remains stationary relative to the screen.
-0.2	Layer -2 units	Moves -12 units on screen; the foreground passes faster.

CAMERA TRAVEL: +10 WORLD UNITS

Distant hills · factor 0.8



World ground · factor 0



Foreground · factor -0.2



X and Y are independent

In a side-scroller, Y Factor = 0 keeps the layer fixed in world Y. If it should move slightly up and down when the camera jumps or follows, try Y = 0.1. Factor is not a speed in units per second.

Check When the camera moves right, a background with factor 0.8 should move less on screen than a layer with factor 0.2.

01 · FOUNDATION

01. Import the package into your project

- 1 Open a **Unity 6000.3.13f1** project in Unity Hub. This is the source target version. If you use another version, check import results and Console errors.
- 2 Copy `Assets/ERAGANX/2D Parallax`, including its **.meta files**, into the destination project's Assets folder. If you received a .unitypackage, import it using **Assets > Import Package > Custom Package**.
- 3 Wait for Unity to finish importing and compiling. Open **Window > General > Console** and confirm there are no red compilation errors.
- 4 Use **File > Save As** to save the scene, for example as `Assets/Scenes/ParallaxPractice.unity`.
- 5 Confirm that **Tools > ERAGANX > 2D Parallax** appears. If it is missing, check the import path and Console.

Use case	Requirements
Core parallax + effects	Runtime assembly: ERAGANX.Parallax2D. The core does not depend on URP, Cinemachine, an external tween library, or the Input System.
Sample HUD + sample player	Unity UI (com.unity.ugui), Input System (com.unity.inputsystem), and TMP Essential Resources. Import TMP resources using Window > TextMeshPro > Import TMP Essential Resources.
URP lights/shadows	URP 17.x, a configured 2D Renderer, and the optional bridge. This project includes URP 17.3.0.
Custom game scripts	UnityEngine and the required package namespaces. If you use a custom asmdef, reference the runtime assembly.

For a new project

You can complete the basic parallax exercise without URP lighting. A core-only distribution with the sample code removed does not need the sample UI dependencies. Set up those dependencies before importing and using the sample scenes.

01 · FOUNDATION

02. Prepare the camera and artwork

Camera setup

- 1 Select **Main Camera** in the Hierarchy. Set **Projection = Orthographic** and Rotation = (0, 0, 0). A typical scene can use camera Z = -10 and artwork Z = 0.
- 2 Start with a value such as **Size = 5**. This is the vertical half-height, so the full view height is 10 world units. The camera looks along +Z at the XY gameplay plane.
- 3 Configure camera movement with your follow script, Cinemachine, or the demo controller. The package does not create camera or player movement by itself.

Sprite import

Setting	Starting choice and reason
Texture Type	Sprite (2D and UI).
Sprite Mode	Single; use Multiple and slicing when needed for a sprite sheet.
Mesh Type	Full Rect for looping layers. Consider transparent padding and edge colours.
Pixels Per Unit	Use a consistent value for the art set, such as 100. Width = pixels / PPU × world scale.
SpriteRenderer Draw Mode	Simple. The infinite scroller does not support Sliced or Tiled geometry.
Scale / rotation	The source sprite's world transform must be axis-aligned, without rotation or shear.
Artwork edges	Left and right edges must match for horizontal repetition. Top and bottom must also match when using Both.

For example, a 1600 × 900 sprite at PPU 100 and scale 1 has a world size of 16 × 9. If Camera Size 5 requires a height of 10, adjust scale, tile size, or the artwork composition. The scroller does not stretch the artwork.

For sprite import fields, see the [Unity 6.3 Sprite texture import reference](#).

Check Select the sprite asset, adjust its Inspector settings, and click Apply. Use Game View to confirm that the artwork covers the screen.

02 · FIRST ENVIRONMENT

03. Create an environment

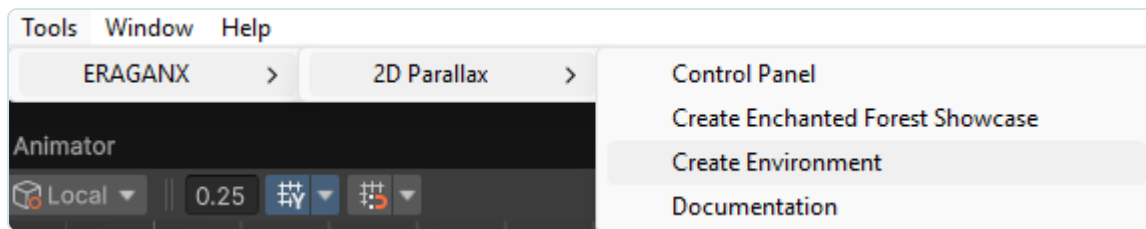


Figure 17 · Choose Create Environment from the Tools menu.

- 1 Click **Tools > ERAGANX > 2D Parallax > Create Environment**.
- 2 The **Parallax Environment** object appears in the Hierarchy and the **Parallax Control Panel** opens. The root object contains the Manager and Camera Adapter.
- 3 Confirm that the panel's **Environment** reference points to your root object. Drag Main Camera into **Camera > Target Camera**.
- 4 To edit an existing environment, select its Manager object or assign it in the open panel's Environment field. Using Create Environment again creates another root object.
- 5 Save the scene. An empty Layers list is normal at this stage; you will add artwork next.

The hierarchy we will build

C# · call example

```

Main Camera
Parallax Environment
  [Parallax2DManager]
  [Parallax2DCameraAdapter]
  Sky / Hills      [SpriteRenderer + Parallax2DLayer]
  Trees            [SpriteRenderer + Parallax2DLayer]
  Near Bushes      [SpriteRenderer + Parallax2DLayer]
  Atmosphere       [optional Camera Anchor]
  Fog / Rays / Particles
  Player + Ground  [your gameplay and physics]
```

Environment and gameplay

Use background layers for artwork. Keep gameplay colliders and the player controller on your game objects. Generated scrolling tiles do not copy colliders or child behaviours.

02 · FIRST ENVIRONMENT

The complete Control Panel layout

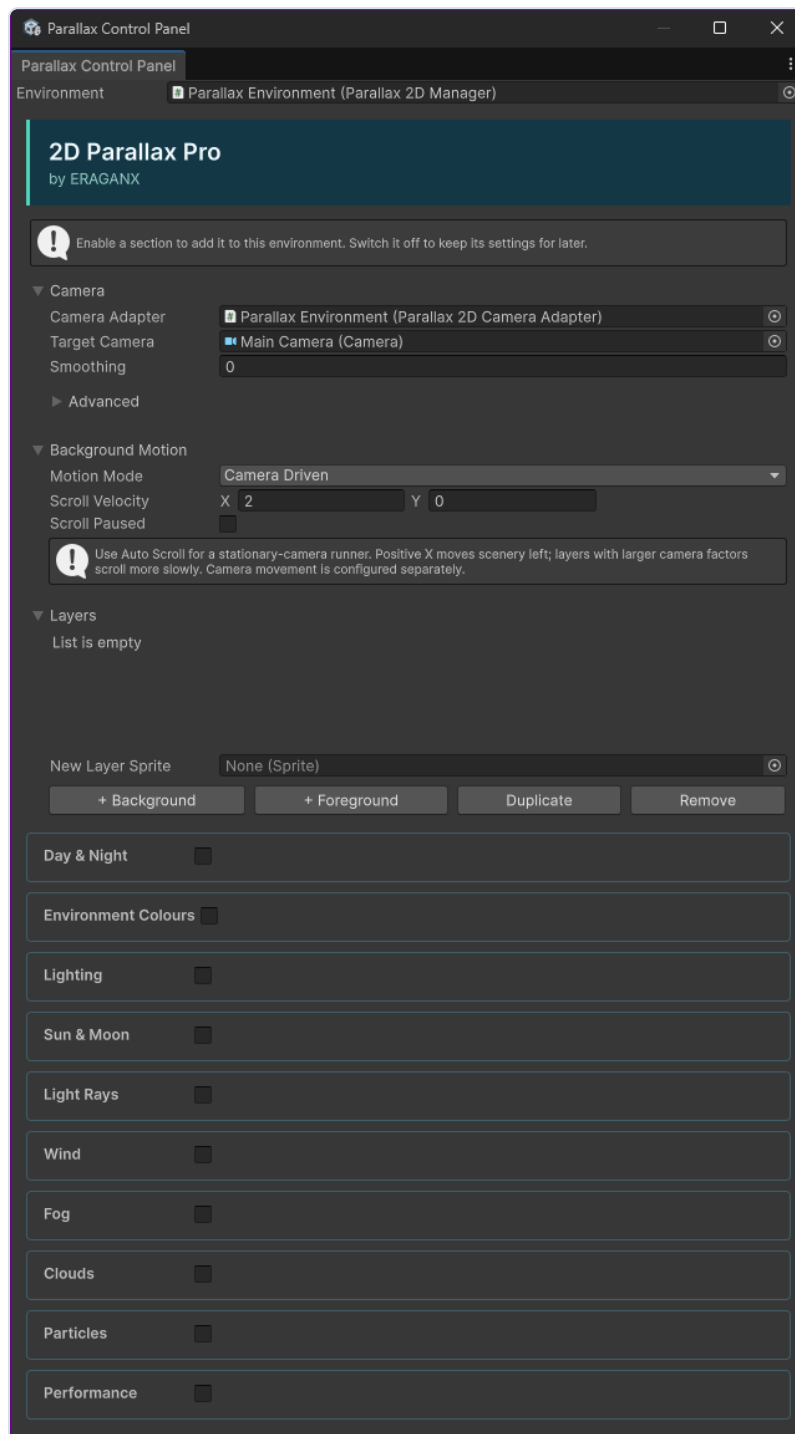


Figure 04 · Camera, motion, layers, and optional sections.

From top to bottom

Environment: the Manager you are editing. Select the correct one when the scene has multiple environments.

Camera: the rendered camera and smoothing of sampled movement.

Background Motion: choose camera-driven motion, independent auto scrolling, or both.

Layers: select a sprite and use + Background or + Foreground. Select a list entry to edit its settings.

Feature checkbox: enables the feature and connects required components and references. Turning it off preserves its settings.

Settings foldout: collapsing it does not stop an enabled effect.

Advanced: additional configuration for that section. There is no global Advanced mode.

Add Another: creates another independent Fog, Clouds, Rays, or Particles effect object.

Edit Mode

Add or remove layers and create new effect objects in Edit Mode. Ordinary runtime Inspector changes are lost when Play Mode ends.

02 · FIRST ENVIRONMENT

04. Camera and Background Motion

Camera field	Use case
Camera Adapter	The component that supplies the rendered camera and sampled movement to the Manager.
Target Camera	The scene's Orthographic Camera. Assign it even for a stationary runner so the tiles cover its view.
Smoothing	0 samples motion and shake directly. Positive values filter the motion; higher values approach the target faster.
Advanced movement source	An IParallaxCameraSource implementation or Position Override. These can separate the camera's actual view from the parallax input.

Motion Mode	When to use it
Camera Driven	A platformer with camera or player follow. Layer positions respond to camera displacement. Scroll Velocity has no effect in this mode.
Auto Scroll	A runner or menu background with a stationary camera. Layer drift follows virtual travel; camera displacement is ignored for parallax.
Camera And Auto Scroll	Combines camera follow with additional automatic scenery scrolling.

```
autoDrift / second = -scrollVelocity * (1 - factor) * autoScrollMultiplier
```

Scroll Velocity is measured in world units per second. X = +6 moves scenery left. Factor X = 0.8 gives -1.2 units/second; factor 0.2 gives -4.8. Y is calculated in the same way.

Three separate pause controls

Scroll Paused stops automatic layer drift. **Camera Paused** stops the sample controller's camera movement and zoom. **Day & Night Paused** stops clock progression. Player movement is controlled separately.

Check For a runner, select Auto Scroll instead of Camera Driven. The background should move even when the camera stays still.

02 · FIRST ENVIRONMENT

05. Add layers and configure depth

- 1 Drag a sprite asset from the Project window into **New Layer Sprite**.
- 2 Click **+ Background** to create a distant layer. Select it in the list and configure Layer Name, Factor, Order, and Tint.
- 3 Assign the next sprite and add another layer. Use **+ Foreground** for a near overlay.
- 4 **Duplicate** copies the selected layer. **Remove** removes it. Unlock a locked layer before editing or removing it.

Layer setting	Result
Layer Name	Use a unique name for preset matching. Choose clear names such as Hills, Trees, and Bushes.
Factor / Offset	Independent X/Y camera response and a world-unit offset from the baseline.
Sorting Layer / Order	Within the same sorting layer, a higher Order draws in front. Placement does not change movement speed.
Tint / Environment Tint Weight	Base colour. Weight 0 ignores environment tint; weight 1 applies the full tint.
Visible / Parallax Enabled	Visible controls rendering visibility. Parallax Enabled controls camera and auto motion; independent effects may continue separately.
Auto Scroll Multiplier	0 excludes this layer from auto drift. Camera Driven response is unaffected.
Clamp Position / Bounds	Restrict world X/Y movement to a rectangle. An unsuitable clamp can stop motion in looping setups.

Example layer	Factor (X,Y)	Order
Distant Hills	(0.85, 0.05)	-100
Far Trees	(0.65, 0.10)	-80
Near Bushes	(0.30, 0)	-20
Foreground Leaves	(-0.10, 0)	10

Check If the player uses Order 0, negative-order backgrounds should appear behind it and Order 10 foreground artwork in front.

02 · FIRST ENVIRONMENT

06. Configure Infinite Scrolling

- 1 Select the layer to repeat in the Layers list. Enable **Infinite Scrolling**; a scroller component is added if needed.
- 2 Choose **Axes = Horizontal**. Vertical and Both repeat sprites along those axes as well.
- 3 Assign the Simple SpriteRenderer to repeat as **Source**. Even if a layer has several renderers, only this source renderer is cloned.
- 4 Start with **Tile Size Override = (0, 0)** and **Spacing = (0, 0)**. Zero size axes use the sprite bounds and world scale.
- 5 Move and zoom the camera in Play Mode. Check seams, missing coverage, and tile-budget warnings in the Console.

Scroller setting	Meaning
Tile Size Override	A nonzero axis specifies tile size in world units. Artwork is not stretched. An incorrect width causes gaps or overlap.
Spacing	The world-unit gap between tiles. Usually 0 for seamless repetition.
Alternate Mirror X / Y	Flips every other tile. This improves joins only when the artwork has suitable symmetry.
Padding Tiles	Extra rows or columns beyond the visible area; start with 1.
Max Tiles	The pool budget for one scroller. Excess coverage requirements produce a source-sprite fallback and warning.
Active / Pooled Tile Count	The numbers of active and pooled tiles. Pooled tiles are reused during play.

Looping artwork ≠ looping gameplay

The scroller does not clone colliders, lights, arbitrary scripts, child hierarchies, or Tilemaps. Use your gameplay system for runner collisions and platform spawning. Camera or source rotation/shear and Sliced/Tiled draw modes are unsupported.

Check After long camera travel and zooming out, screen edges should remain covered, without lines or empty gaps between tiles.

02 · FIRST ENVIRONMENT

Guided exercise: your first parallax scene

Build a scene with correct camera response, factors, and sorting before adding effects. You can use sample artwork or four layers of your own.

- 1 Set Main Camera to Orthographic, Size 5, position (0, 0, -10), and rotation (0, 0, 0). Assign it as the Environment Target Camera.
- 2 Add Distant Hills, Far Trees, and Near Bushes using + Background. Add Foreground Leaves using + Foreground. Use the earlier factor/order table.
- 3 Adjust sprite world scale and position to cover the camera view. Enable Horizontal Infinite Scrolling for tileable art.
- 4 Set **Background Motion = Camera Driven**. Keep optional effects off initially and save the scene.
- 5 In Play Mode, slowly change camera X from 0 to 10, or attach the test script below to Main Camera.
- 6 Confirm that distant layers move less on screen and the foreground passes faster. Stop Play Mode and save the scene.

GuideCameraMove.cs

```
using UnityEngine;

public sealed class GuideCameraMove : MonoBehaviour
{
    public Vector2 velocity = new Vector2(1f, 0f);

    void LateUpdate()
    {
        transform.position +=
            (Vector3)(velocity * Time.deltaTime);
    }
}
```

Attach the script

Create `Assets/Game/Scripts/GuideCameraMove.cs`. After compilation, drag it onto Main Camera. Disable or remove this test movement script when using your production follow controller.

Check Hills at factor 0.85 and Trees at 0.65 should move differently. If they appear to move at the same rate, check the selected layer factors and Target Camera.

07. The Day & Night clock

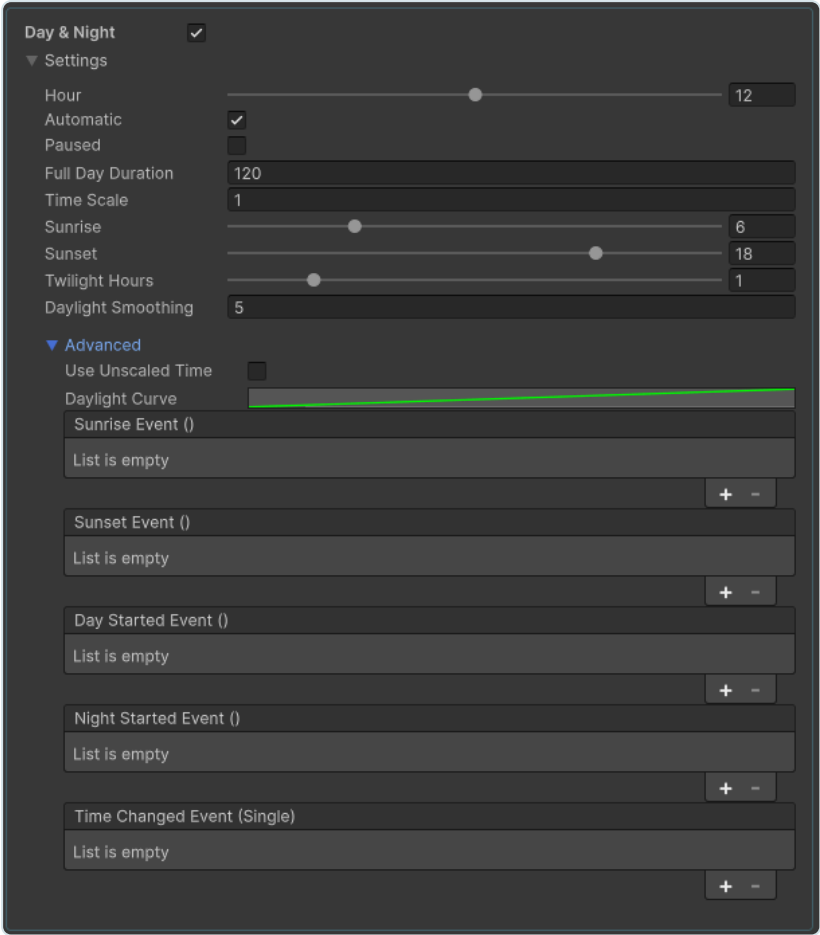


Figure 05 · Clock settings and Inspector UnityEvents.

Field	How to configure it
Hour	0 = midnight, 6 = morning, 12 = noon, and 18 = evening. 24 wraps back to 0.
Automatic / Paused	Automatic enables clock progression. Paused stops progression while keeping the current hour.
Full Day Duration / Time Scale	120 seconds at Scale 1 gives a 120-second full cycle. At Scale 2, it takes 60 seconds.
Sunrise / Sunset / Twilight	Here: 6 / 18 / 1 clock hour. Daylight fades after sunrise and before sunset.
Daylight Smoothing	Use a value such as 5 for visual fading. Higher values respond faster; 0 applies immediately.

Check Turn Automatic off and switch Hour between 12 and 0. The clock alone does not change artwork colours; enable Environment Colours too.

03 · DAY, LIGHT & ATMOSPHERE

Clock timing, curves, and events

```
clockHours / second = 24 / fullDayDuration * timeScale
```

With **Automatic** on and **Paused** off, the clock advances in Update. Time Scale = 0 stops it; a negative value runs it backward. Full Day Duration is measured in game seconds.

Advanced setting	Use case
Use Unscaled Time	The clock and its visual smoothing use real elapsed time even when Time.timeScale = 0. This does not make every other effect use unscaled time.
Daylight Curve	Shapes the daylight amount from 0 to 1, without changing clock speed. Tune morning and evening lighting and effect responses.
Sunrise / Sunset Event	Invoked when the configured boundary is crossed.
Day / Night Started Event	Invoked when a daytime or nighttime phase begins.
Time Changed Event (Single)	Passes the current hour to a UI or game script as a float argument.

Bind an Inspector event

- 1 Under Day & Night > Advanced, click + in the required Event list.
- 2 Drag the scene object containing your public callback method into the object field.
- 3 Choose a parameterless method in the Function menu. For Time Changed, choose a **Dynamic float** method; a static float selection sends a fixed number.
- 4 In Play Mode, move the hour across sunrise or sunset and check the callback. A C# event subscription example appears in the API section.

External clock / transition

If a saved-game clock or environment transition owns time, turn Automatic off or pause it. Two systems writing time simultaneously produce unexpected results. SetTime(hour, false) suppresses all clock events.

SetTime advances by the signed difference from the current hour to the requested wrapped hour. To advance three hours across midnight, use AdvanceHours(3). Each boundary event fires at most once per update, even for a multi-day skip.

03 · DAY, LIGHT & ATMOSPHERE

08. Environment Colours

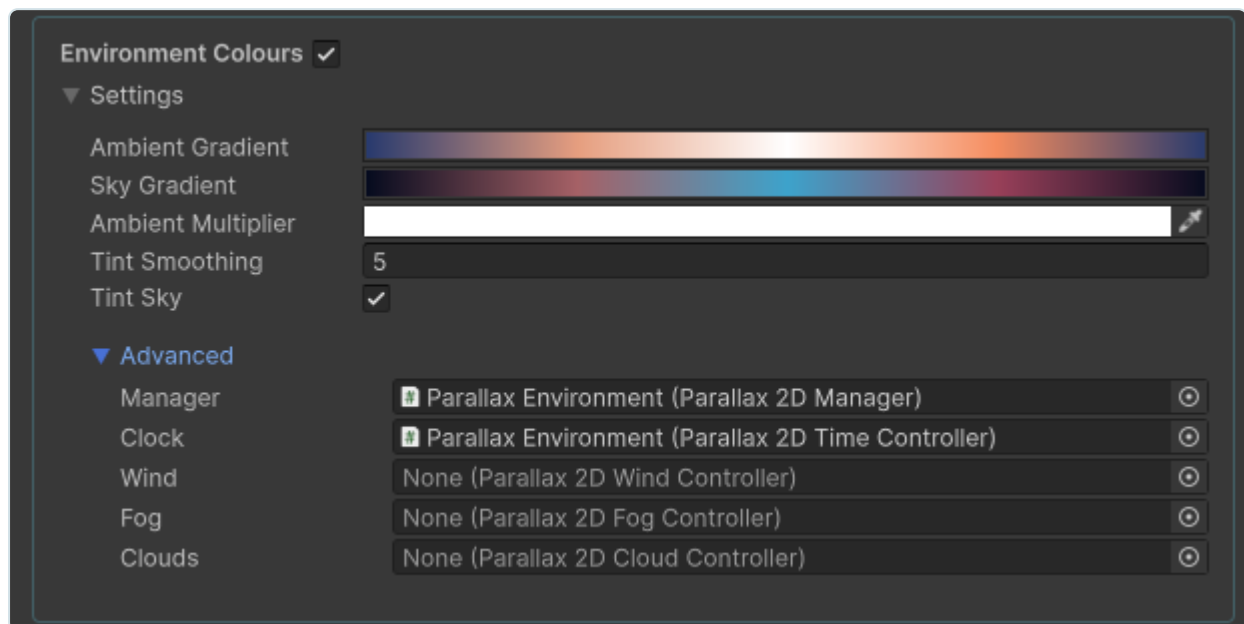


Figure 06 · Connects layer tint and the camera sky gradient to the clock.

- 1 Enable **Day & Night** and **Environment Colours**. Under Advanced, check that **Manager** and **Clock** belong to the correct environment.
- 2 Click **Ambient Gradient** and add time-of-day colour keys. 0 / 1 = midnight, 0.25 = 06:00, 0.5 = noon, and 0.75 = 18:00.
- 3 Configure **Sky Gradient** for the camera background colour. Turn **Tint Sky** on.
- 4 With **Ambient Multiplier** set to White, the original gradient colours are preserved. Another colour multiplies the gradient.
- 5 Set **Tint Smoothing** = 5 and scrub time. Use each layer's **Environment Tint Weight** to reduce tint on UI-like or independently coloured layers.

Colour recipe: dark blue midnight, warm sunrise, bright neutral noon, and orange/pink sunset. Matching colours at gradient endpoints 0 and 1 avoid a midnight seam.

When the sky change is not visible

Tint Sky changes camera.backgroundColor. A fullscreen sky sprite can cover it. Control that sky sprite through its layer's ambient tint or a separately authored colour controller.

Check Check layers and sky at Hour 0/6/12/18. If the result stays dark with a White multiplier and layer weight 1, check the authored sprite tint and alpha.

03 · DAY, LIGHT & ATMOSPHERE

09. Lighting modes and intensity

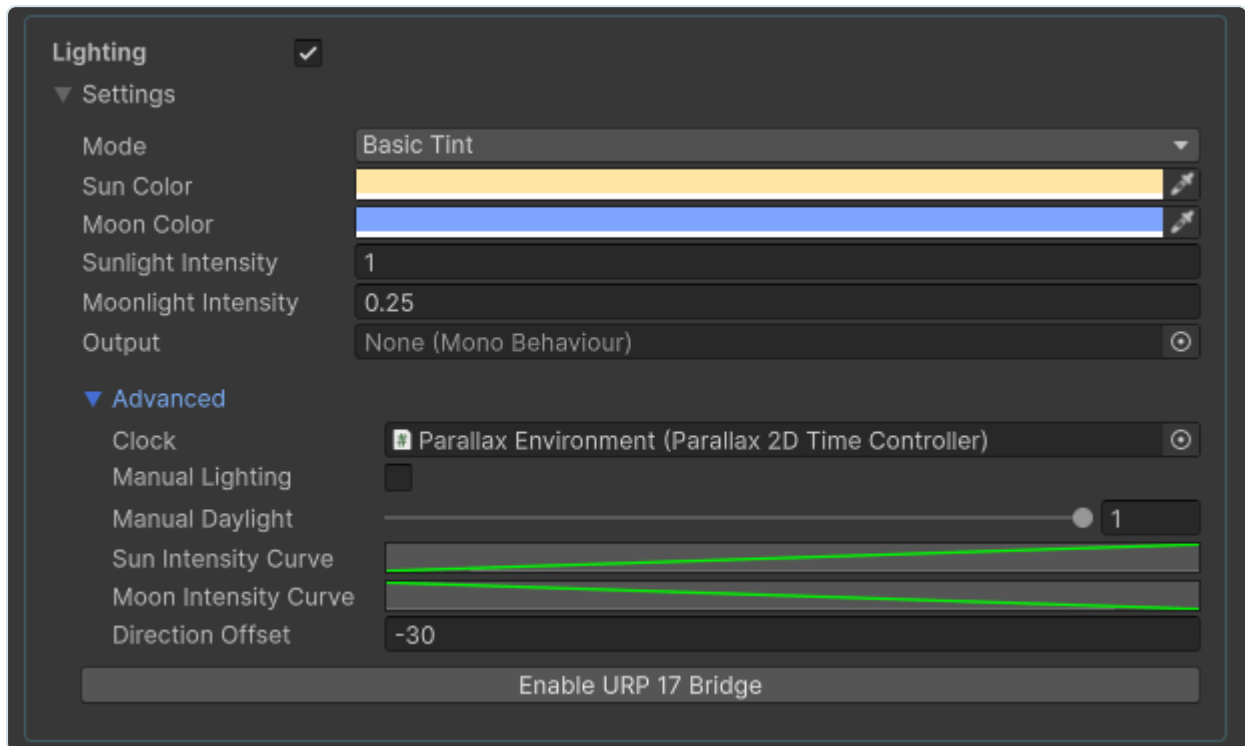


Figure 07 · The Lighting section: clock, curves, and optional output bridge.

Mode / Field	Behaviour
Basic Tint	Disables atmospheric ray support. Enable Environment Colours for layer and sky colours; the Lighting controller alone does not tint sprites.
Lightweight	Supplies sun/moon intensity data for rays. URP Light2D and shadows are not required.
URP2D	Sends intensity, colour, and direction to the Output bridge. Requires a configured URP 2D Renderer and authored lights.
Sun / Moon Color	Output light colours. Visible Sun & Moon sprite colours have separate settings.
Sunlight / Moonlight Intensity	Base brightness, shown here as 1 and 0.25. Curves multiply it according to daylight.
Manual Lighting / Daylight	Uses manual daylight from 0 to 1 instead of clock daylight. Time-based Direction still uses the clock.
Intensity Curves / Direction Offset	The curve X axis is daylight from 0 to 1. Direction Offset adds degrees to the daily rotation.

Check Select Lightweight or URP2D to display rays. Basic Tint prevents rays from rendering even when the Rays component is enabled.

03 · DAY, LIGHT & ATMOSPHERE

Optional: URP 17 lights and shadows

Follow these steps only when you need Light2D illumination or authored 2D shadows. Core parallax does not require the URP bridge. This project includes URP 17.3.0; the package bridge targets 17.x.

- 1 Confirm **Universal RP 17.x** is installed in Package Manager. Configure a **2D Renderer** on the active pipeline asset and assign the appropriate Graphics/Quality settings.
- 2 Click **Control Panel > Lighting > Enable URP 17 Bridge**. The package creates an active bridge assembly from its included `.asmdef.txt` template. Wait for compilation.
- 3 Add **Parallax2DURPLightBridge** to an environment child object. Assign authored **Light2D** components to Sun, Moon, and optional Ambient.
- 4 Drag the bridge component into **Lighting > Output** and choose **Mode = URP2D**.
- 5 Use **Sprite-Lit-Default** or a compatible Sprite Lit material on sprites that should receive light. The supplied unlit materials do not respond to Light2D.
- 6 Author **ShadowCaster2D** shapes for silhouettes that should cast shadows, and assign them to the bridge arrays. Assign `CompositeShadowCaster2D` groups if needed.
- 7 Check Light2D affected sorting layers, shadow settings, and range. Compare colour, intensity, and shadows at Clock Hour 12 and 0 in Game View.

Sun sprites and lights are separate

The Sun & Moon section uses `SpriteRenderers` for visible artwork. The URP bridge Sun/Moon fields need `Light2D` components. Make sure you drag the correct component from the object.

Before removing URP, remove bridge components and the generated

`Integrations/URP/ERAGANX.Parallax2D.URP.asmdef`, keeping the `.txt` template. You do not need to define `ERAGANX_HAS_URP17` globally. The source does not claim URP 18+ compatibility.

Unity reference: [2D Renderer Data](#) and [Prepare sprites for 2D lighting](#).

03 · DAY, LIGHT & ATMOSPHERE

10. Sun & Moon visuals

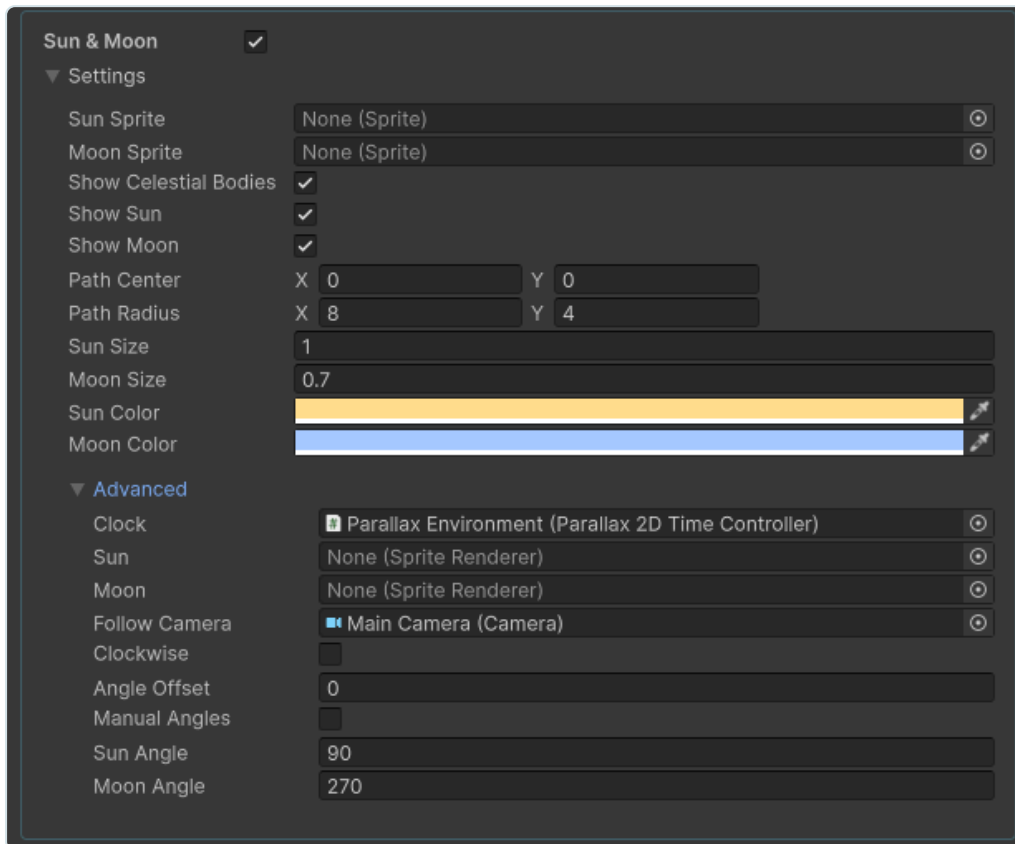


Figure 08 · Sun/Moon sprites, elliptical paths, and camera-relative positioning.

Field	Purpose
Sun / Moon Sprite	Assigning an asset creates a visual child SpriteRenderer if missing. None means there is no visible body.
Show Celestial Bodies / Sun / Moon	Shows or hides the group or an individual body. The clock and other effects work independently.
Path Center / Radius	Centre offset and horizontal/vertical ellipse radii in world units. The example radius is (8,4).
Sun / Moon Size, Color	Local scale and sprite tint. Final world size also depends on sprite PPU and artwork dimensions.
Follow Camera	Anchors the path to the assigned camera XY. Without a camera, it uses the controller's world position.
Clockwise / Angle Offset	Reverses travel direction and offsets the path angle in degrees.
Manual Angles / Sun, Moon Angle	Uses manual angles instead of time-based positioning. An assigned clock can still control opacity and fading.

Check At noon, the sun should be high on its path; at night, the moon should be visible. If a body is missing, check its sprite, alpha, visibility, radius, camera view, and sorting order.

03 · DAY, LIGHT & ATMOSPHERE

11. Light Rays - screenshot walkthrough

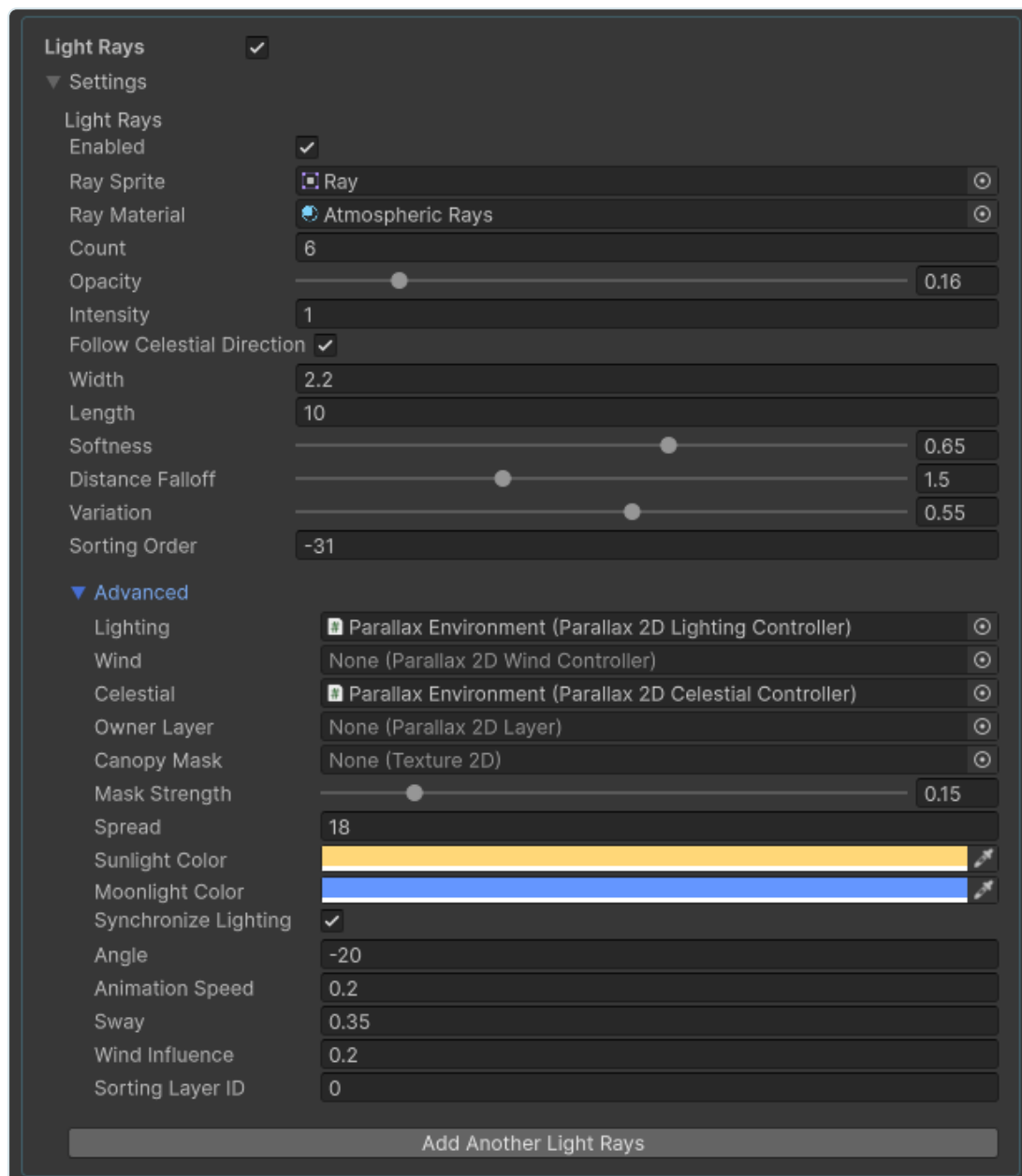


Figure 09 · Settings for soft atmospheric light shafts.

- 1 Choose **Lighting = Lightweight** or URP2D and enable Light Rays. Assign **Ray Sprite** and the optional **Ray Material**.
- 2 Start with Count 6, Opacity 0.16, Width 2.2, and Length 10. Draw foreground silhouettes in front of the rays.
- 3 Turn **Synchronize Lighting** on for clock-driven intensity. If you have Sun & Moon visuals, use **Follow Celestial Direction**.

03 · DAY, LIGHT & ATMOSPHERE

Light Rays - what the settings mean

Field	Result
Count / Opacity / Intensity	Number of shafts, base alpha, and brightness multiplier. Quality settings can cap Count.
Width / Length / Spread	Base local size of each ray and the group's horizontal distribution width.
Softness / Distance Falloff	Material-supported soft edges and fading away from the emitter. Higher falloff shortens the visible shaft.
Variation	Deterministic spacing, size, and brightness variation. 0 produces uniform beams.
Sorting Layer ID / Order	Use a lower order than foreground tree silhouettes so they draw in front of the rays. -31 is an example.
Lighting / Wind / Celestial	Component references for intensity synchronization, wind sway, and the visible sun/moon direction.
Owner Layer	Allows that layer's Visible and Light Rays Enabled settings to control ray visibility.
Canopy Mask / Mask Strength	A texture where white passes light and black blocks it. Strength 0 ignores it. Requires a compatible ray material.
Synchronize Lighting	On blends ray colour and alpha using computed sun/moon intensities. Off uses a constant Sunlight Color source.
Follow Celestial Direction / Angle	Uses the visible celestial body's direction. Falls back to Angle when no body is assigned or the option is off.
Animation Speed / Sway	Animation phase speed and angular sway in degrees. Speed 0 freezes the phase.
Wind Influence	Additional angular sway from the assigned Wind Controller.

Mask limitations

The canopy mask is a ray-local atmospheric approximation. It does not produce geometry-based or physically correct leaf shadows. Foreground occlusion is sorting-based. See the URP setup for authored 2D shadows.

Check If rays are too bright, reduce opacity or intensity. Check that synchronized alpha decreases with zero sun intensity and that moon intensity can produce night rays.

03 · DAY, LIGHT & ATMOSPHERE

12. Wind - bend visual children

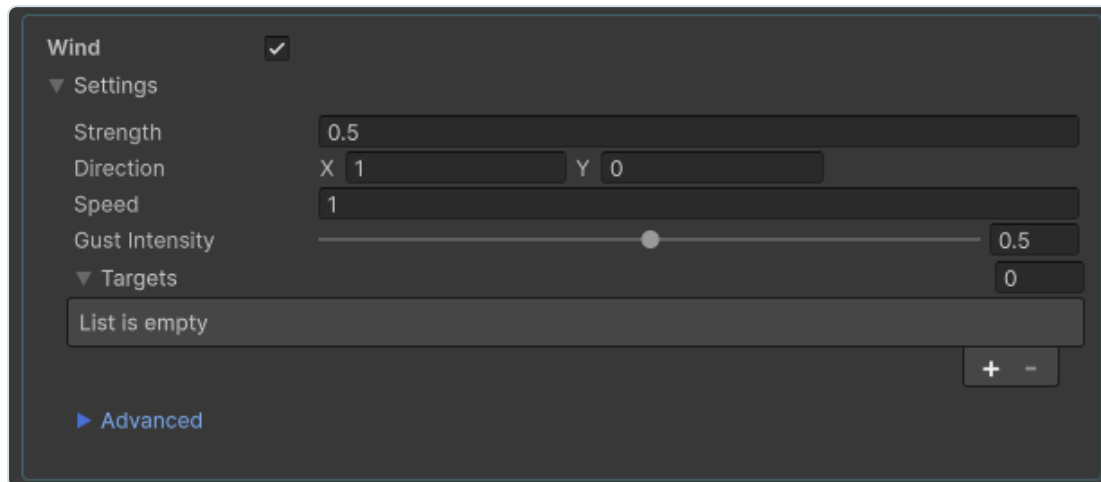


Figure 10 · An empty Targets list means no wind-bent visuals have been assigned.

- 1 Enable Wind. Start with Strength 0.5, Direction (1,0), Speed 1, and Gust Intensity 0.5.
- 2 Place grass or tree artwork on a **visual child Transform of the layer root**. Use a pivot suitable for rotational bending.
- 3 Click **Targets > +** and drag the child Transform into **Visual**. Configure Target Multiplier and Bend Degrees, such as 3.
- 4 In Play Mode, change Strength between 0 and 1 and inspect the sway. Use the layer's Wind Multiplier to tune that layer's response.

Field	Meaning
Strength	Base wind amount for all targets. 0 removes bending.
Direction	The normalized X component controls bending direction; (0,1) produces no X bending.
Speed	Sway/gust phase speed. 0 freezes the phase.
Gust Intensity	Periodic strength variation; 0 steady base strength.
Targets	Visual, target multiplier, and bend angle. Runtime bending skips a target that is a layer root.

How wind works

Wind uses Transform rotation sway, not mesh deformation or automatic physics wind. The infinite scroller does not clone tile hierarchies, so do not assume source-child sway is copied to children of every tile.

Check Assign a valid child Transform in Targets. Disabling wind restores the visual's baseline rotation.

03 · DAY, LIGHT & ATMOSPHERE

13. Fog - density and noise mist

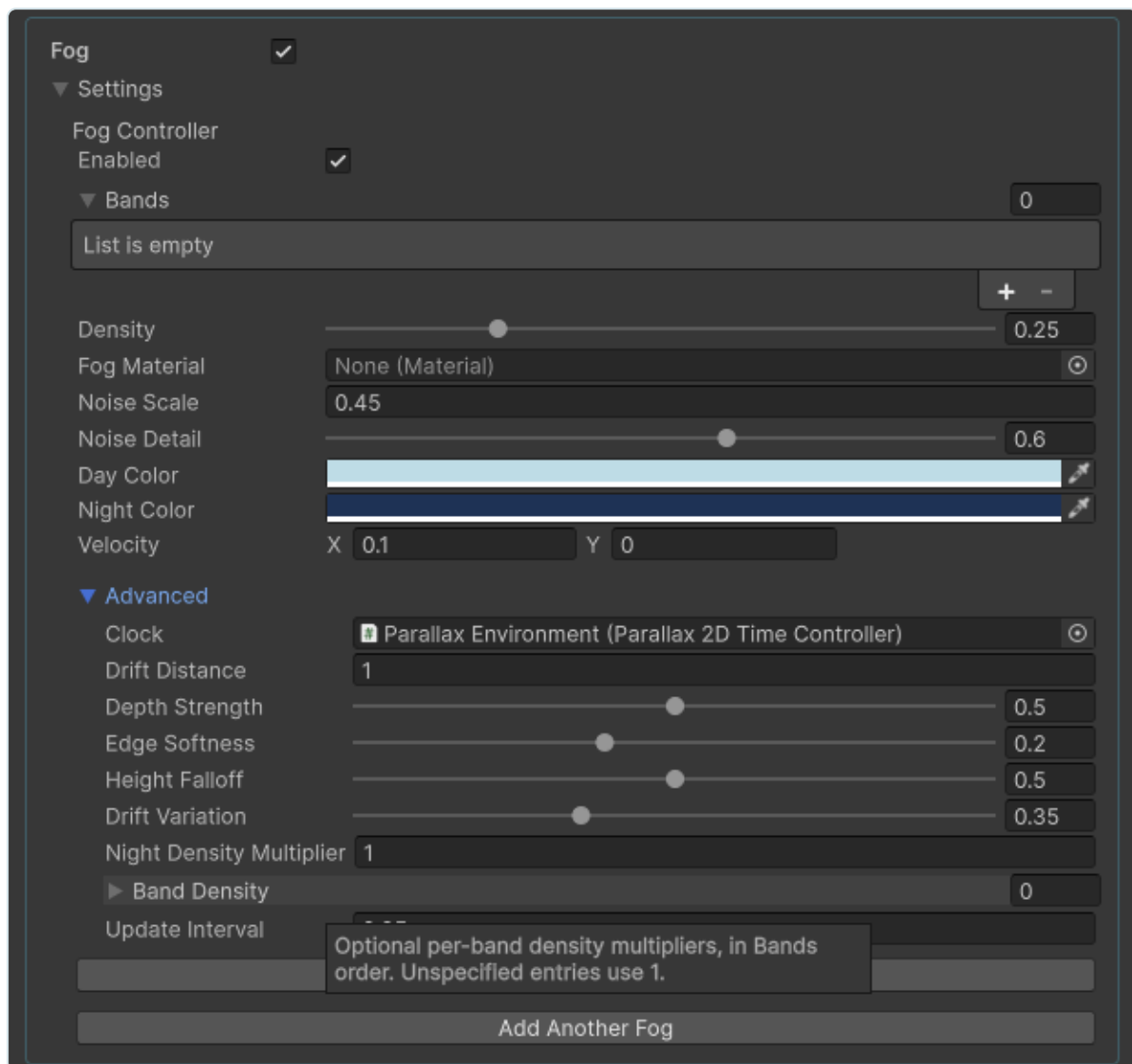


Figure 11 · Fog bands, colour, material, and per-band configuration.

- 1 Enable Fog and use **Add Fog Band**. This creates an empty child SpriteRenderer and adds it to the Bands array.
- 2 Select Fog Band in the Hierarchy, assign a transparent fog sprite, and adjust its position, scale, and sorting order for the scene.
- 3 Start with Density 0.25 and Velocity (0.1,0). For animated noise, assign the supplied Atmospheric Fog material.

The screenshot has Bands = 0

Enabling the controller alone does not create fog artwork. Fog appears after you add valid SpriteRenderers to Bands and configure their sprites, alpha, and materials.

03 · DAY, LIGHT & ATMOSPHERE

Fog bands and Advanced settings

Setting	Meaning
Density / Day, Night Color	Base opacity and colour blending using clock. VisualDaylight. Alpha also affects the result.
Fog Material	Optional atmospheric noise material. None retains authored sprite-only fog.
Noise Scale / Detail	Noise size/frequency and fine detail in a compatible fog material.
Velocity / Drift Distance	Bands move in bounded oscillation; the material noise also scrolls. This is not endless sprite travel.
Depth Strength	Density variation by band order. Higher values make earlier bands more transparent.
Edge Softness / Height Falloff	Material-supported sprite edge softening and vertical opacity fading.
Drift Variation	Motion speed and phase variation between bands.
Night Density Multiplier	1 keeps the same base density; higher values make night fog thicker.
Band Density	Per-band multipliers in Bands array order. Unspecified entries use 1.
Update Interval	Seconds between fog-band position updates. Quality may override this; it does not pause the whole effect.

Build a fog group with depth

- 1 Create 2-3 SpriteRenderers for near and far fog. Vary width, height, and Order to place them between layers.
- 2 Keep fog bands as standalone atmosphere children, **outside Parallax2DLayer objects**. Avoid the Manager tint and fog tint writing to the same renderer colour.
- 3 For camera-relative fog, add **Parallax2DCameraAnchor** to the parent Atmosphere object and assign the Environment Camera Adapter. The anchor follows camera XY and retains the existing Z.
- 4 For another independent density or material, use **Add Another Fog** and author its Bands separately.

Check Test Hour 0 and 12, camera panning, and fog disable/re-enable. If fog is missing, check band sprites, renderer state, alpha, sorting, and coverage.

03 · DAY, LIGHT & ATMOSPHERE

14. Clouds - drift and optional shadow

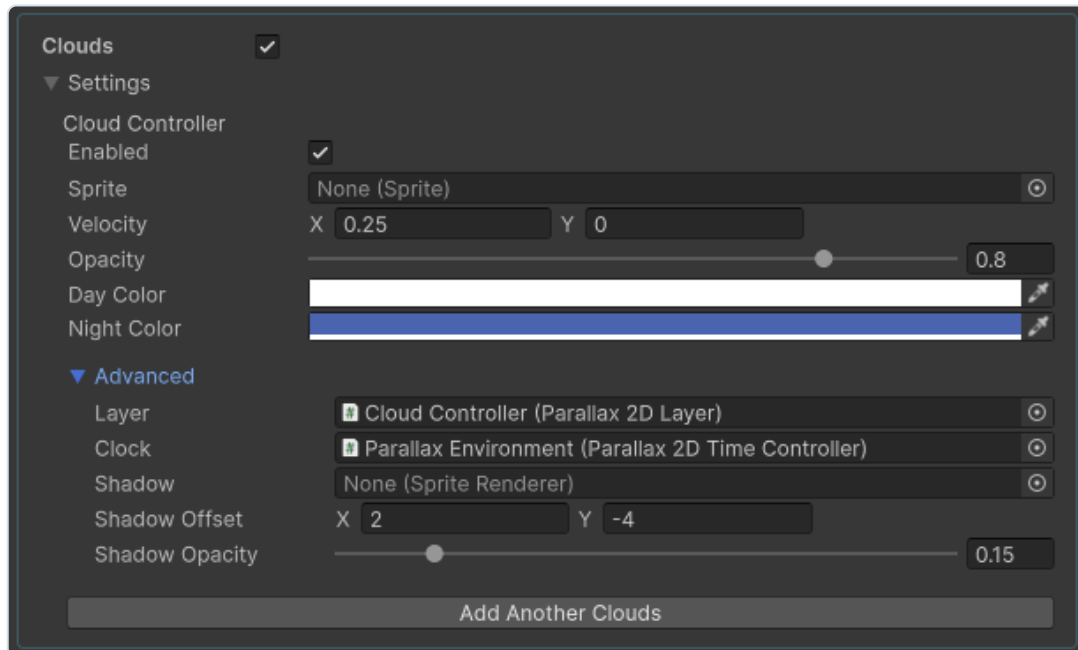


Figure 12 · The cloud controller adds independent drift to its parallax layer.

- 1 Enable Clouds and assign a cloud sprite asset to **Sprite**.
- 2 Check the **Layer** reference. Configure its factor/order and Horizontal Infinite Scrolling if needed.
- 3 Set **Velocity** = **(0.25,0)** and **Opacity** = **0.8**. Positive cloud X drift moves right; the Manager's positive auto-scroll X convention moves scenery left.
- 4 Assign Day/Night Color and Clock. Cloud velocity adds drift on top of Camera Driven motion.

Advanced field	Result
Shadow	Optional separately authored transparent silhouette SpriteRenderer. Assigning the Shadow field alone does not create geometry.
Shadow Offset	X/Y offset added to the authored shadow's local position.
Shadow Opacity	Daytime multiplier. The shadow fades out at night.

Use **Add Another Clouds** for another layer/controller with a different sprite, speed, or opacity.

The cloud controller owns layer.tint

An enabled cloud controller captures the original tint and applies day/night colour × opacity. If your code or a transition also writes layer.tint, consider the ownership conflict. Cloud shadows are a visual approximation.

Check Clouds should drift even with a stationary camera. Enable looping for tileable cloud art and test it over an extended period.

03 · DAY, LIGHT & ATMOSPHERE

15. Particles - fireflies, leaves, dust

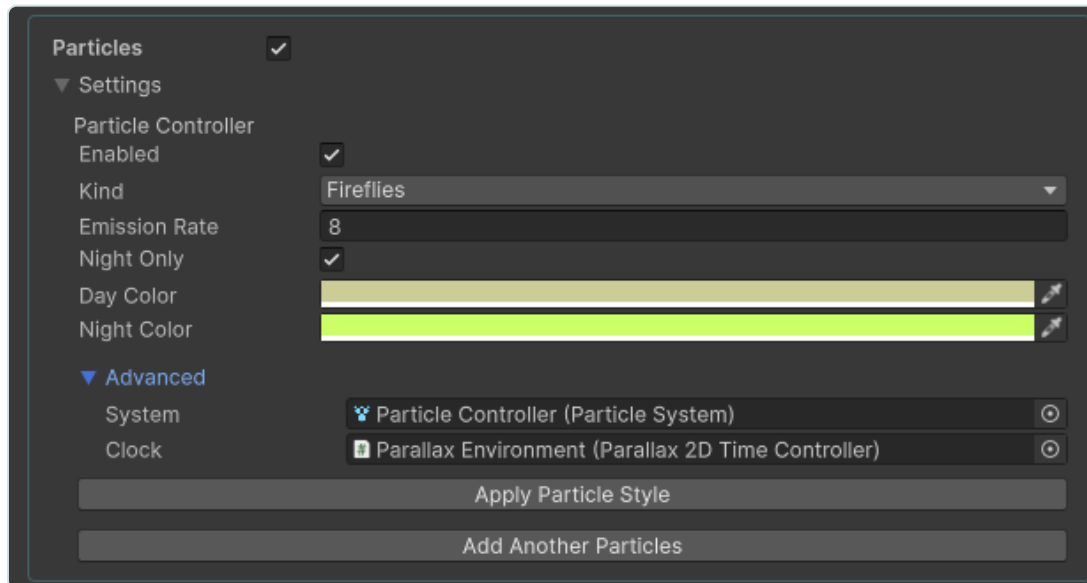


Figure 13 · After changing Kind, click Apply Particle Style.

- 1 Enable Particles. The required ParticleSystem/controller object is created. Check Advanced > System.
- 2 Choose Fireflies, Leaves, Dust, or Floating as **Kind**. Click **Apply Particle Style** to configure the modules.
- 3 Start with a value such as **Emission Rate = 8**. This is particles per second before quality and night multipliers.
- 4 For Fireflies, turn **Night Only** on and test at Clock Hour 0. Emission fading during daylight is expected.
- 5 Adjust the ParticleSystem Renderer material, sorting, and shape/scale to suit Game View. Use Add Another Particles when needed.

Setting	Behaviour
Night Only	Emission $\times$ (1 - VisualDaylight). Without a clock, daylight = 1, so night-only emission is 0.
Day / Night Color	Colour blending for new particles. Controls the system's main.startColor.
Apply Particle Style	Lifetime, size, speed, shape, and velocity defaults. Fireflies sets Night Only true; other kinds set it false.

After configuring the style

Clicking the style button again may overwrite manually tuned modules. Simulation is in world space, so existing particles stay in world space even with a camera-relative emitter.

Check If fireflies are missing, check Hour, Night Only, emission, renderer material/alpha, camera view, and the Quality multiplier.

04 · PRESETS & SAMPLES

16. Performance and quality budgets

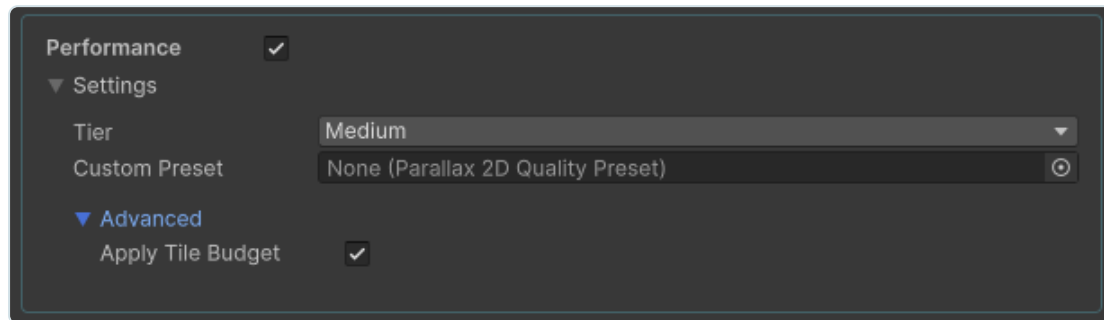


Figure 14 · Quality Tier, Custom Preset, and Apply Tile Budget.

Budget	Low	Medium	High
Max rays / emitter	3	8	24
Particle multiplier	0.25	0.60	1.00
Fog density multiplier	0.40	0.75	1.00
Fog position update interval	0.15 s	0.075 s	0.033 s
Tile cap / scroller	128	256	512

These are the built-in tier values in the source. An assigned **Custom Preset** overrides the selected Tier's budgets. With **Apply Tile Budget** on, the lower of authored Max Tiles and the tier cap is used.

- 1 Start with Medium and test the scene's intended camera size, aspect ratio, and zoom range.
- 2 Switch to Low and check reduced rays/particles, fog alpha and position cadence, and tile coverage. Quality also changes fog density, which affects the visual mood.
- 3 After adding or removing effects/scrollers at runtime, call `quality.Refresh()` to update cached collections.
- 4 Use the Unity Profiler in a target-device build to inspect CPU/GPU cost, batches, memory, and particles. A tier name does not guarantee FPS.

Prioritize coverage

If a reduced tile cap causes source-fallback warnings, consider larger sprite tiles, less zoom-out/padding, or a larger authored budget/custom preset. Pool count is not the number of layers.

Check After changing values in a Custom Preset, call `quality.Apply()` at runtime. `SetQuality(tier)` clears `customPreset` and selects a built-in tier.

04 · PRESETS & SAMPLES

17. Create preset assets

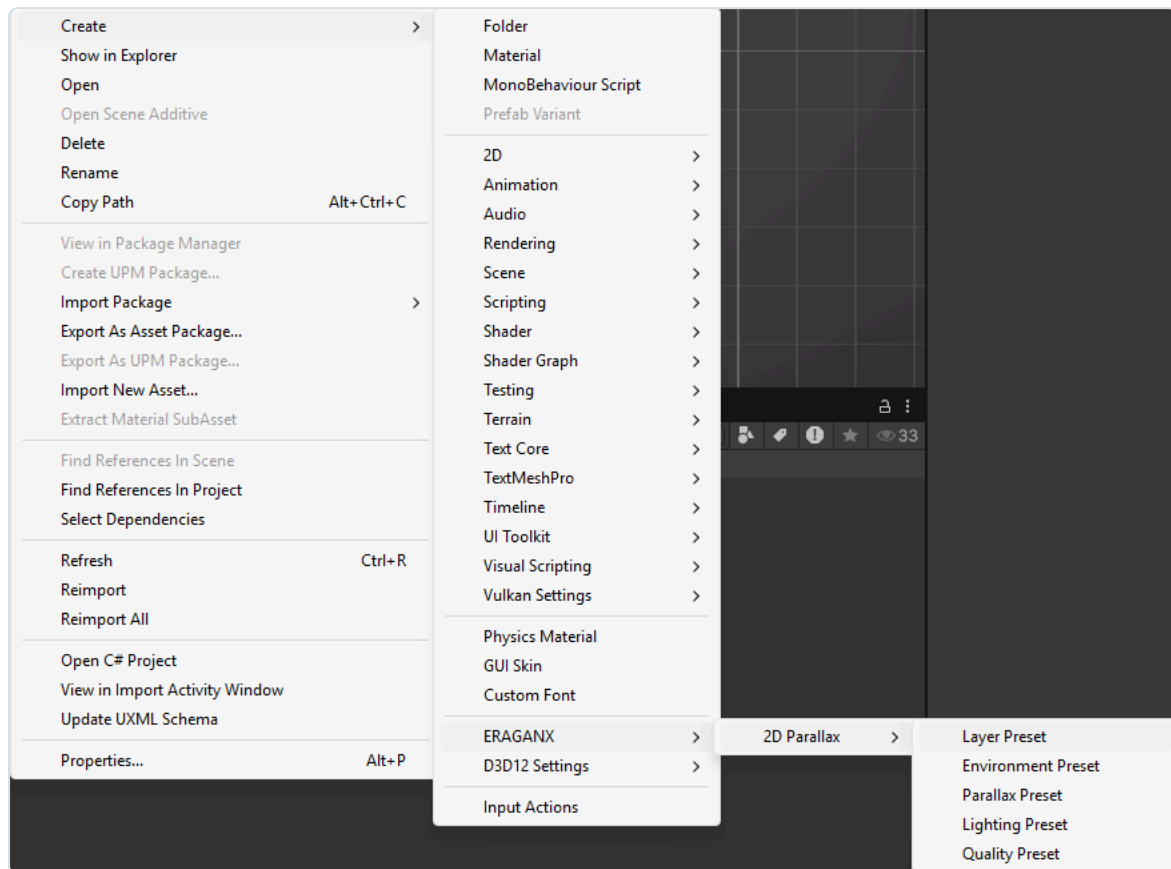


Figure 18 · Project window > Create > ERAGANX > 2D Parallax.

Preset type	Stored settings
Layer Preset	One layer's name, placement, factor, offset, tint, visibility, sorting, looping, tile size, and spacing.
Environment Preset	Layer configurations, start hour/day duration, ambient/sky gradients, ambient multiplier, wind/fog/cloud velocity.
Lighting Preset	Mode, sun/moon colours, intensity, and direction offset.
Quality Preset	Ray/particle/fog/tile budgets.
Parallax Preset	References to Environment, Lighting, and Quality presets in one composite asset.

Right-click your preset folder in the Project window and choose an asset type from the menu above. Configure it in the Inspector. Apply it using `ApplyTo(...)` or sample runtime controls that call the API. The screenshot menu is for preset **asset creation**.

Preset matching

Environment preset layers match by **Layer Name**; use unique names. Existing sprite artwork and transform geometry are retained. Missing layers are created only with `createMissingLayers: true`. Presets do not swap/delete existing art or clone complete scenes.

04 · PRESETS & SAMPLES

18. Demo player and camera controller

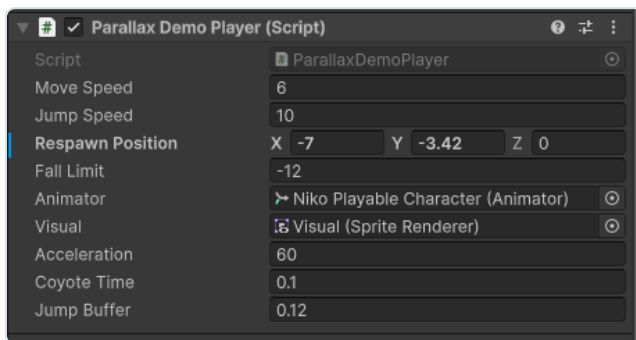


Figure 15 · ParallaxDemoPlayer Inspector.

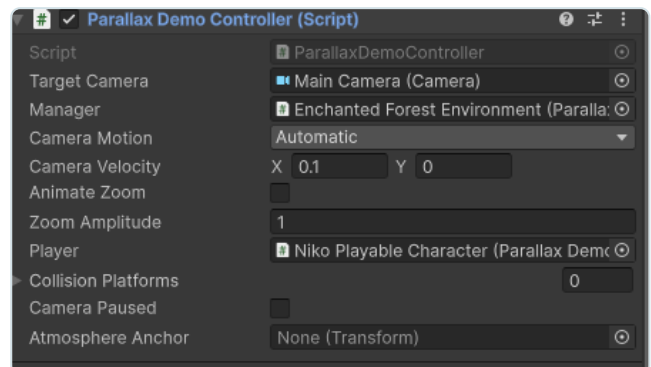


Figure 16 · ParallaxDemoController Inspector.

Player field	Meaning
Move / Jump Speed	Maximum horizontal speed and upward jump velocity in world units/second.
Acceleration / Coyote Time	Horizontal acceleration and the brief jump window after leaving the ground, in seconds.
Jump Buffer	Seconds to retain a jump request while waiting to land.
Respawn Position / Fall Limit	Falling below Fall Limit Y sends the player to the respawn position.
Animator / Visual	Animator with Speed, VerticalSpeed, and Grounded parameters, plus the SpriteRenderer used to flip facing direction.

Camera Motion	Use case
Automatic	Follows an assigned player; otherwise moves at Camera Velocity.
Stationary	Leaves camera position and zoom under your controller's control.
Follow Player	Follows only an assigned player. With no player, the camera does not move.
Automatic Movement	Moves at Camera Velocity even with a player assigned.

For a runner

Set Demo Camera Motion = Stationary and Manager Background Motion = Auto Scroll. Camera Paused does not stop background drift or player input. Collision Platforms and Atmosphere Anchor are sample setup fields; you can use your own game controller.

04 · PRESETS & SAMPLES

Learn with the sample scenes

In the Project window, double-click a scene under [Assets/ERAGANX/2D Parallax/Samples/Sample](#) and enter Play Mode. The top-left HUD in samples 1-7 provides runtime preview controls, separate from the Editor Control Panel.

Scene	What to study
1 - Basic Parallax	Four simple layers, moving camera, factors, and sorting.
2 - Infinite Scrolling	Horizontal/vertical loops, diagonal movement, zoom, and teleport coverage.
3 - Dynamic Day Night	Clock, sun/moon, gradient colours, Hour, and pause controls.
4 - Forest Lighting	Soft shafts and direction/intensity synchronization. URP shadows need optional authored integration.
5 - Dynamic Environment	Clouds, fog, wind-bent grass, fireflies, and runtime presets.
6 - Mobile Performance	Cartoon scene, Low quality default, and live layer/quality information. Measure real cost with a Profiler/device build.
7 - Complete Platformer Demo	Niko player, movement/jump, collisions, following camera, full effects, and respawn.
8 - Enchanted Forest	Painted layers, mist, canopy rays, URP lights/shadows, and glows. Composed for a bounded scene rather than infinite looping.

- 1 Test the available Hour, Day Cycle, Rays, Fog, Camera, Quality, Preset, Teleport, and Reset controls. Disabled modules display N/A or Not enabled.
- 2 Samples 6-7 support A/D or arrow keys and Space; focus Game View. On-screen buttons support touch and pointer input.
- 3 To edit the HUD, open **Sample Controls Canvas > Controls Panel** in Edit Mode. The shared prefab is [Samples/Shared/UI/ParallaxDemoControls.prefab](#). Scene 8 does not have this HUD.

Sample reset

ResetDemo restores camera/player/platform geometry and calls `Manager.ResetEnvironment`. It is not a general save system that resets all clock, preset, and game state. Author persistent environments in Edit Mode.

05 · RUNTIME API

19. Get started with the C# API

The package API consists of C# methods, public configuration fields, and events on Unity components. No HTTP endpoints or API keys are needed. There is no singleton, so keep references to the environment you want to control.

C# · call example

```
using UnityEngine;
using ERAGANX.Parallax2D.Core;           // enums, interfaces
using ERAGANX.Parallax2D.Runtime;        // manager, presets
using ERAGANX.Parallax2D.Layers;         // layer
using ERAGANX.Parallax2D.Scrolling;       // tile scroller
using ERAGANX.Parallax2D.Environment;    // clock, colours
using ERAGANX.Parallax2D.Lighting;       // lights, sun/moon
using ERAGANX.Parallax2D.Effects;        // wind, fog, clouds
```

- 1 Create `Assets/Game/Scripts` and a `.cs` file whose filename matches the example's class name.
- 2 Paste the code and save it. After Unity compiles, attach the script to an empty **GameEnvironmentControls** scene object.
- 3 Drag Manager and the required Clock, Lighting, Fog, Cloud, and Quality references into the Inspector. Scripts with public fields follow on the next pages.
- 4 If your scripts use a custom assembly definition, add a reference to **ERAGANX.Parallax2D**. Sample-only classes also need the Samples assembly and UI/Input dependencies.
- 5 Enter Play Mode. Call the methods from your gameplay code. Public void methods can be bound to UI Button OnClick or UnityEvents.

Methods and fields

Methods such as `SetSpeed` and `SetTime` sanitize input; direct fields provide configuration. Keep direct assignments finite. Null references or disabled objects/effects may produce no output. An API call does not automatically enable the controller.

Check Confirm there are no namespace/assembly errors in the Console and that script Inspector fields reference valid scene components.

05 · RUNTIME API

API example 1: create a runtime layer

Attach the script to a fresh environment and assign Manager, renderedCamera, and tileableSprite. The sprite must be tileable. For an existing authored layer, use its reference instead of adding it again.

GuideRuntimeLayers.cs

```
using UnityEngine;
using ERAGANX.Parallax2D.Core;
using ERAGANX.Parallax2D.Runtime;
using ERAGANX.Parallax2D.Layers;

public sealed class GuideRuntimeLayers : MonoBehaviour
{
    public Parallax2DManager manager;
    public Camera renderedCamera;
    public Sprite tileableSprite;
    public Parallax2DQualityController quality;
    private Parallax2DLayer hills;

    void Start()
    {
        if (!manager || !renderedCamera || !tileableSprite)
            return;
        manager.SetCamera(renderedCamera);
        hills = manager.AddLayer(tileableSprite,
            "Runtime Hills", LayerPlacement.Background);
        hills.SetSpeed(new Vector2(0.8f, 0.1f));
        hills.sortingOrder = -80;
        hills.SetLooping(LoopAxes.Horizontal);
        hills.SetVisible(true);
        if (quality) quality.Refresh();
    }

    public void ShowHills(bool show)
    {
        if (hills) hills.SetVisible(show);
    }

    public void RemoveHills()
    {
        if (manager && hills)
            manager.RemoveLayer(hills, destroyObject: true);
        hills = null;
        if (quality) quality.Refresh();
    }
}
```

Runtime structure

AddLayer creates a SpriteRenderer and Layer as a Manager child. RemoveLayer defaults to destroyObject = false, retaining the object while disabling/unregistering the layer. RefreshLayers can register a retained child again. Destruction with true applies in Play Mode.

05 · RUNTIME API

API example 2: endless runner background

Keep the rendered camera stationary. Enable Horizontal Infinite Scrolling on background layers. Set Auto Scroll Multiplier = 0 on ground/platform gameplay layers that must stay still.

GuideEndlessRunner.cs

```
using UnityEngine;
using ERAGANX.Parallax2D.Core;
using ERAGANX.Parallax2D.Runtime;
using ERAGANX.Parallax2D.Layers;

public sealed class GuideEndlessRunner : MonoBehaviour
{
    public Parallax2DManager manager;
    public Camera renderedCamera;
    public Parallax2DLayer groundLayer;
    public float runSpeed = 6f;

    void Start()
    {
        if (!manager || !renderedCamera) return;
        manager.SetCamera(renderedCamera);
        manager.SetMotionMode(ParallaxMotionMode.AutoScroll);
        manager.SetScrollVelocity(new Vector2(runSpeed, 0f));
        manager.scrollPaused = false;
        if (groundLayer) groundLayer.autoScrollMultiplier = 0f;
    }

    public void SetRunSpeed(float speed)
    {
        runSpeed = Mathf.Max(0f, speed);
        if (manager)
            manager.SetScrollVelocity(new Vector2(runSpeed, 0f));
    }

    public void SetScrollPaused(bool paused)
    {
        if (manager) manager.scrollPaused = paused;
    }
}
```

runSpeed = 6, multiplier = 1

Layer travel / second

Factor X 0.85 -0.9 world units

Factor X 0.65 -2.1 world units

Factor X 0.30 -4.2 world units

Ground multiplier 0 0 auto-scroll units

Check Bind a background-pause button to SetScrollPaused(true). The background should stop while the player and jump controller remain independent.

05 · RUNTIME API

API example 3: clock calls and events

Assign Clock in the Inspector. Pair event subscriptions in OnEnable with unsubscriptions in OnDisable. Hour updates can be frequent, so you do not need to log every update.

GuideClockEvents.cs

```
using UnityEngine;
using ERAGANX.Parallax2D.Environment;

public sealed class GuideClockEvents : MonoBehaviour
{
    public Parallax2DTimeController clock;
    public float DisplayHour { get; private set; }

    void OnEnable()
    {
        if (!clock) return;
        DisplayHour = clock.Hour;
        clock.OnSunrise += HandleSunrise;
        clock.OnNightStarted += HandleNight;
        clock.OnTimeChanged += HandleTime;
    }

    void OnDisable()
    {
        if (!clock) return;
        clock.OnSunrise -= HandleSunrise;
        clock.OnNightStarted -= HandleNight;
        clock.OnTimeChanged -= HandleTime;
    }

    void HandleSunrise() { Debug.Log("Sunrise"); }
    void HandleNight() { Debug.Log("Night started"); }
    void HandleTime(float hour) { DisplayHour = hour; }

    public void SetNight()
    {
        if (!clock) return;
        clock.automatic = false;
        clock.SetTime(0f);
    }

    public void ResumeCycle()
    {
        if (!clock) return;
        clock.automatic = true;
        clock.Resume();
    }
}
```

Select SetNight() or ResumeCycle() under UI Button > OnClick. Use `clock.AdvanceHours(3)` to advance game hours rather than seconds, or `clock.SetTime(savedHour, false)` to load saved time without events.

05 · RUNTIME API

API example 4: weather, lighting, and effects

Enable the relevant sections in Edit Mode and assign their references. `Environment.ApplyWeather` changes only its assigned Wind/Fog/Cloud references. For multiple effects, explicitly loop over the required components.

GuideAtmosphere.cs

```
using UnityEngine;
using ERAGANX.Parallax2D.Environment;
using ERAGANX.Parallax2D.Lighting;
using ERAGANX.Parallax2D.Effects;

public sealed class GuideAtmosphere : MonoBehaviour
{
    public Parallax2DEnvironmentController environment;
    public Parallax2DLightingController lighting;
    public Parallax2DCloudController clouds;
    public Parallax2DParticleController particles;
    public Parallax2DCelestialController celestial;

    public void SetStormMood()
    {
        if (environment)
            environment.ApplyWeather(1.2f, 0.65f, 0.9f);
        if (lighting)
        {
            lighting.SetSunlightIntensity(0.5f);
            lighting.SetMoonlightIntensity(0.3f);
        }
        if (clouds) clouds.velocity = new Vector2(0.6f, 0f);
    }

    public void SetClearMood()
    {
        if (environment)
            environment.ApplyWeather(0.25f, 0.1f, 0.3f);
        if (lighting) lighting.SetSunlightIntensity(1f);
    }

    public void UseLeaves()
    {
        if (!particles) return;
        particles.kind = AmbientParticleKind.Leaves;
        particles.Configure();
        particles.emissionRate = 5f;
    }

    public void ShowSunAndMoon(bool show)
    {
        if (celestial) celestial.SetVisible(show);
    }
}
```

This changes existing effect settings. Fog density is a direct field: `fog.enabled = true; fog.density = 0.4f;`. For clouds: `clouds.opacity = 0.8f;`. Assign fog bands and cloud sprites beforehand.

05 · RUNTIME API

API example 5: apply and capture presets

Create Layer, Environment, Lighting, and Parallax preset assets in the Project window and assign them to the script fields. Cancel an active transition before applying instantly. Set up ownership if an external script controls the clock.

GuidePresets.cs

```
using UnityEngine;
using ERAGANX.Parallax2D.Runtime;
using ERAGANX.Parallax2D.Layers;

public sealed class GuidePresets : MonoBehaviour
{
    public Parallax2DManager manager;
    public Parallax2DLayer selectedLayer;
    public Parallax2DLayerPreset layerPreset;
    public Parallax2DEnvironmentPreset environmentPreset;
    public Parallax2DPreset compositePreset;
    public Parallax2DQualityController quality;
    private Parallax2DEnvironmentPreset sessionSnapshot;

    public void ApplyLayer()
    {
        if (layerPreset && selectedLayer)
            layerPreset.ApplyTo(selectedLayer);
    }

    public void ApplyEnvironment()
    {
        if (!environmentPreset || !manager) return;
        environmentPreset.ApplyTo(manager,
            createMissingLayers: false);
        if (quality) quality.Refresh();
    }

    public void ApplyComposite()
    {
        if (compositePreset && manager)
            compositePreset.ApplyTo(manager);
    }

    public void CaptureSession()
    {
        if (!manager) return;
        if (!sessionSnapshot)
            sessionSnapshot = ScriptableObject
                .CreateInstance<Parallax2DEnvironmentPreset>();
        sessionSnapshot.CaptureFrom(manager);
    }

    void OnDestroy()
    {
        if (sessionSnapshot) Destroy(sessionSnapshot);
    }
}
```

CaptureSession creates an in-memory runtime snapshot; it does not save an .asset file to disk. Persistent capture needs an Editor tooling/AssetDatabase workflow. Presets do not replace existing sprite artwork.

05 · RUNTIME API

API example 6: smooth environment transition

Add **Parallax2DTransition** to the environment root and assign Manager. Create a Night Environment Preset with startHour, colours, fog, wind, and matching layer configurations.

GuideTransition.cs

```
using UnityEngine;
using ERAGANX.Parallax2D.Runtime;
using ERAGANX.Parallax2D.Environment;

public sealed class GuideTransition : MonoBehaviour
{
    public Parallax2DTransition transition;
    public Parallax2DEnvironmentPreset nightPreset;
    public Parallax2DTimeController clock;
    private bool ownsClock;
    private bool savedAutomatic;

    public void BeginNight()
    {
        if (!transition || !nightPreset || !clock) return;
        if (!ownsClock) savedAutomatic = clock.automatic;
        ownsClock = true;
        clock.automatic = false;
        transition.TransitionTo(nightPreset, 3f,
            applyDiscreteSettingsOnComplete: true);
    }

    void Update()
    {
        if (ownsClock &&
            (!transition || !transition.IsTransitioning))
            RestoreClock();
    }

    public void StopTransition()
    {
        if (transition) transition.Cancel();
        RestoreClock();
    }

    void RestoreClock()
    {
        if (ownsClock && clock)
            clock.automatic = savedAutomatic;
        ownsClock = false;
    }

    void OnDisable()
    {
        if (ownsClock) StopTransition();
    }
}
```

Assign the same Clock used under the transition's Manager. Bind the button to BeginNight(). Completion or cancellation restores the previous Automatic state; you control the Paused state separately.

05 · RUNTIME API

What transitions change and their limits

Settings	TransitionTo behaviour
Factor / Offset / Tint	Smoothly interpolates numeric values on layers matched by Layer Name. Use unique names.
Hour	Takes the shortest path around the 24-hour clock. Intermediate SetTime calls suppress events. Stop automatic clock progression first.
Ambient multiplier	Smooth colour blend. The ambient and sky gradients themselves are not tweened between presets.
Wind / Fog / Cloud velocity	Interpolates strength, density, and velocity on the first matching components, not every effect entry.
Discrete settings = false	Does not apply gradients, loop axes, sorting, visibility, or day duration. The default is false.
Discrete settings = true	Calls destination.ApplyTo(manager) on completion to apply preset settings. Its clock SetTime event behaviour also applies.
Sprite references / transforms	Does not swap existing artwork, create hierarchies/colliders, or tween artwork transforms.
New TransitionTo / Cancel	A new call starts from current values. Cancel stops at current values rather than reverting to the first preset.

One owner for each value

An enabled cloud controller rewrites layer.tint, which a transition can also change. Plan colour control for that layer separately. Gameplay scripts, timelines, and automatic clocks can conflict when they write the same values simultaneously.

Transitions on a pause screen

C# · call example

```
transition.useUnscaledTime = true;
// The transition can progress while Time.timeScale is 0.
// For clock-controlled visual fades during the same pause:
clock.useUnscaledTime = true;
```

Unscaled transition progression does not make all effect animation and drift use unscaled time. Environment Tint Smoothing and other controllers use their own deltaTime, so test the visual behaviour you need on a pause screen.

Check Start a night transition, then cancel or replace it halfway through. A replacement should start from the current state without jumping back to the original values.

05 · RUNTIME API

API example 7: quality, reset, camera switch

GuideRuntimeControls.cs

```

using UnityEngine;
using ERAGANX.Parallax2D.Core;
using ERAGANX.Parallax2D.Runtime;

public sealed class GuideRuntimeControls : MonoBehaviour
{
    public Parallax2DManager manager;
    public Parallax2DQualityController quality;
    public Parallax2DQualityPreset customQuality;
    public Camera alternateCamera;

    public void LowQuality()
    {
        if (quality) quality.SetQuality(QualityTier.Low);
    }

    public void HighQuality()
    {
        if (quality) quality.SetQuality(QualityTier.High);
    }

    public void ApplyCustomQuality()
    {
        if (!quality || !customQuality) return;
        quality.customPreset = customQuality;
        quality.Apply();
    }

    public void SwitchCamera()
    {
        if (manager && alternateCamera)
            manager.SetCamera(alternateCamera,
                              preservePositions: true);
    }

    public void RebaseAfterOriginShift()
    {
        if (manager) manager.Rebaseline();
    }

    public void ResetLayerGeometry()
    {
        if (manager) manager.ResetEnvironment();
    }
}

```

ResetEnvironment ≠ full game reset

Resets layer geometry and drift, then rebases against the current camera. It does not reset the clock, presets, particles, player, or save state. Switching the Manager camera does not activate/deactivate rendered cameras; your camera system handles that.

For a floating origin, shift camera, player, gameplay, and environment geometry together, then call `Rebaseline()`. Unity float coordinates do not provide unlimited precision.

05 · RUNTIME API

Advanced API: custom camera movement source

Implement **IParallaxCameraSource** to exclude camera shake from background parallax or supply another world position as input. Keep the rendered Camera assigned for tile coverage.

GuideCameraSource.cs

```
using UnityEngine;
using ERAGANX.Parallax2D.Core;

public sealed class GuideCameraSource : MonoBehaviour,
    IParallaxCameraSource
{
    public Transform unshakenCameraTarget;

    public Vector3 CameraPosition => unshakenCameraTarget
        ? unshakenCameraTarget.position
        : transform.position;
}
```

- 1 Add the script to a scene object and assign unshakenCameraTarget.
- 2 Drag this component into Environment Camera Adapter > Advanced > **Movement Source**. Keep **Override Position** off; when on, its value takes priority over the source.
- 3 Keep the actual rendered Orthographic Camera assigned as **Target Camera**. A custom source does not replace the camera.
- 4 Check coverage during shake and zoom. CameraFrame.Position drives parallax; ViewPosition is the actual camera position.

When manual evaluation is needed

The Manager normally samples the camera and calls AdvanceScroll/Evaluate in LateUpdate at execution order **10000**. If your camera controller runs later, first consider fixing its order. With a manual driver, disable the Manager's automatic LateUpdate and advance only once per frame.

C# · call example

```
// Inside your manual LateUpdate driver after camera movement:
// manager.enabled = false; disables its automatic LateUpdate.
if (manager.cameraAdapter.TryGetFrame(Time.deltaTime,
    out var frame))
{
    manager.AdvanceScroll(Time.deltaTime);
    manager.Evaluate(frame);
}
```

No extra update calls are needed

Do not call Evaluate, Tick, or Advance again from your Update in the default setup. Time or drift can advance twice. Use the intended setters for normal gameplay.

06 · API REFERENCE

API quick reference: Manager, Layer, Camera

API	Use case
Manager.Layers / HasCamera	Read-only layer registry and status indicating a valid orthographic Manager camera.
AddLayer(Sprite, string, LayerPlacement)	Creates and returns a child SpriteRenderer + Parallax2DLayer. Defaults: name "Layer", placement Background.
RemoveLayer(layer, destroyObject=false)	Disables and unregisters. Explicit true destroys the GameObject in Play Mode.
RefreshLayers()	Rediscovered and registers owned child layers. A retained removed child may register again.
SetCamera(camera, preservePositions=true)	Assigns the adapter and rebases. False resets geometry before rebasing.
SetMotionMode(mode) / SetScrollVelocity(Vector2)	CameraDriven / AutoScroll / CameraAndAutoScroll. Velocity is sanitized to finite values.
scrollPaused / AdvanceScroll(float)	Auto-drift pause field and manual advancement in seconds. Normal LateUpdate already advances.
Rebaseline() / ResetEnvironment()	Captures the current geometry/camera reference or resets layer geometry and drift.
AssignAutomaticSorting(bg=-100, fg=10, step=1)	Assigns background/foreground sorting orders in registered-list order.
Evaluate(CameraFrame)	Evaluates layer position/appearance and enabled scrollers.
Layer.SetSpeed(Vector2) / SetVisible(bool)	Factor and rendering visibility. SetSpeed replaces invalid axes with zero.
Layer.SetLooping(LoopAxes)	None / Horizontal / Vertical / Both. Adds a missing scroller but does not enable an existing disabled scroller.
Layer.Rebaseline(frame) / ResetToBaseline()	Captures the layer baseline or resets its geometry and drift.
SetEnvironmentTint(Color) / RefreshRenderers()	Applies the environment multiplier or recaptures authored renderer state.
AddDrift(Vector2) / ResetDrift()	Adds or clears independent drift. With enabled looping, drift wraps to the tile period.
CameraAdapter.SetCamera / ResetFilter	Assigns the camera or clears smoothing state.
TryGetFrame(float, out CameraFrame)	Gets a valid orthographic frame. Returns false for a missing or perspective camera.

Scroller fields: source, axes, tileSizeOverride, spacing, alternateMirrorX/Y, paddingTiles, maxTiles. Outputs: TileSize, VisibleBounds, ActiveTileCount, PooledTileCount. ReleaseTiles() releases generated tiles. Call RefreshRenderers deliberately if runtime renderer values are already tinted.

06 · API REFERENCE

API quick reference: Clock, Environment, Lighting

API	Use case
Clock.Hour / NormalizedTime	Read-only wrapped hour from 0 to 24 and hour/24.
IsDay / Daylight / VisualDaylight	Day flag, curve-based daylight, smoothed visual daylight.
SetTime(float, emitEvents=true)	Wraps a finite requested hour and advances by the difference. False suppresses all events.
AdvanceHours(double, emitEvents=true)	Adds signed game hours; supports midnight and multi-day movement. Automatic/Paused flags do not block it.
Advance(float seconds)	Advances according to day duration/time scale when Automatic is true and Paused is false.
Pause() / Resume() / SetTimeScale(float)	Pauses/resumes automatic clock advancement and sets its speed multiplier.
EvaluateVisuals(float deltaTime)	Manually evaluates visual daylight smoothing. Normal Update already does this.
OnSunrise / OnSunset	C# Action events. Each boundary fires at most once per update.
OnDayStarted / OnNightStarted	C# phase-start events. During reverse time, the phase follows the direction of travel.
OnTimeChanged	Action<float> with the current wrapped hour. No event for zero delta or emitEvents false.
Environment.ApplyWeather(wind,fog,cloudCover)	Sanitizes wind to nonnegative and fog/cloudCover to 0..1, then updates assigned references.
Environment.OnWeatherChanged	Action<float,float,float> sanitized weather values callback.
Environment.Evaluate(float)	Updates ambient layer tint and optional camera sky colour. Requires a clock.
Lighting.SetSunlightIntensity(float)	Nonnegative base sunlight value.
Lighting.SetMoonlightIntensity(float)	Nonnegative base moonlight value.
SunIntensity / MoonIntensity / Direction	Computed outputs based on curves/daylight. Direction is in degrees.
SupportsRays / mode / output	BasicTint disables rays. URP2D uses a compatible IParallaxLightOutput.
Celestial.SetVisible(bool) / Evaluate()	Shows/hides the Sun/Moon group or updates position and tint.

Interfaces

`IParallaxWeatherReceiver.ApplyWeather(float,float,float)` supports custom weather systems. `IParallaxLightOutput.ApplyLighting(Color,float,Color,float,float)` and `RestoreLighting()` support custom light output. The assigned output `MonoBehaviour` must implement the interface and be active.

06 · API REFERENCE

API quick reference: Effects, Quality, Presets

API / configuration	Use case
Wind.SetStrength(float) / CurrentStrength	Nonnegative base strength and current gust-adjusted amount. Direction, speed, and targets are public fields.
Fog.enabled / density / bands	Component toggle, base density, and standalone SpriteRenderer array. There is no SetDensity method.
Clouds.velocity / opacity / shadow	Drift speed, layer alpha, and optional shadow reference. There are no SetVelocity/SetOpacity methods.
Particles.Configure() / kind	Applies selected AmbientParticleKind style modules. The Kind enum is in the Effects namespace.
Particles.emissionRate / nightOnly	Authored emission and night fading. Set your desired values after Configure.
Effects.Evaluate(...)	Wind/Fog/Clouds/Rays use deltaTime; Particle Evaluate has no parameters. Normal lifecycle calls are already present.
Quality.SetQuality(QualityTier)	Low/Medium/High. Clears customPreset and applies the tier.
Quality.customPreset / Apply()	Assigns and applies a custom quality asset. Call Apply again after editing its values.
Quality.Refresh()	Rediscovered cached effect/scroller lists and applies current quality.
LayerPreset.CaptureFrom / ApplyTo	Captures or applies one layer configuration.
EnvironmentPreset.CaptureFrom(manager)	Captures layer/time/colours and first wind/fog/cloud values. Not a full scene snapshot.
EnvironmentPreset.ApplyTo(manager, false)	Updates matching layers. True creates missing layers; existing layers are not deleted.
LightingPreset.ApplyTo(lighting)	Mode/colours/base intensity/direction apply.
ParallaxPreset.ApplyTo(manager)	Applies settings from Environment, Lighting, and Quality references together.
Transition.TransitionTo(preset, seconds, false)	Numeric environment transition. True also applies discrete preset settings on completion.
Transition.IsTransitioning / Cancel()	Current status and cancellation that preserves current values.
Transition.Tick(float)	Manual advancement hook. Normal Update already calls it.
CameraAnchor.cameraAdapter / offset / Evaluate()	Makes a standalone atmosphere group follow actual camera XY.

Distinguish an enabled=false controller from an inactive GameObject. Setters change configuration without creating missing sprites, bands, or materials. Disabling Quality restores captured budgets/multipliers. ApplyWeather, presets, or transitions may select only one component when several effects exist.

07 · TROUBLESHOOTING

Troubleshooting: setup and movement

Problem	What to check or fix
Tools menu is missing	Check Console compilation errors, package-folder import, and the Editor assembly. For namespace errors, configure your script asmdef reference.
Layer is not visible	Check sprite asset, SpriteRenderer enabled, Visible, tint alpha, camera clipping/culling mask, position/scale, and sorting order.
Camera moves but no parallax	Check a valid orthographic Target Camera, active Manager/layer, and Parallax Enabled. Auto Scroll intentionally ignores camera displacement.
All layers appear to move alike	Set different X/Y factors. Placement does not change speed. Remember that factor 1 is stationary relative to the screen.
Stationary runner background stays still	Check Auto Scroll mode, nonzero Scroll Velocity and multiplier, Scroll Paused false, and a valid Manager camera frame.
Ground moves during auto scroll	Set groundLayer.autoScrollMultiplier = 0. Confirm the camera is stationary and the collider/gameplay setup is separate.
Scenery moves after camera pause	Camera Paused controls the sample camera. Set Manager.scrollPaused true separately.
Tile seams / black lines	Check tileable edges, Full Rect, source bounds/PPU/world scale, texture padding/filter/compression, and Spacing 0. Mirroring is not an automatic seam fix.
Zoom-out gaps or budget warning	Check whether coverage exceeds Max Tiles. Try larger tiles, authored/custom budgets, or less padding/zoom-out. Quality tile caps also apply.
Loop geometry does not match	Check Simple SpriteRenderer, no world rotation/shear, correct Source, and Tile Size Override. Pivot and PPU are preserved.
Removed runtime layer returns	RemoveLayer(false) retains the object. RefreshLayers rediscovers it. Use true for permanent runtime removal.
Camera switch / teleport jump	Check movement ownership and camera execution order. SetCamera preserves positions by default. Shift geometry together for origin changes, then Rebaseline.

Check To isolate a problem, turn optional effects off and test the camera with one layer first. Enable looping and effects one at a time afterward.

07 · TROUBLESHOOTING

Troubleshooting: effects and runtime API

Problem	What to check or fix
Changing Hour does not change colours	Check Environment Colours enabled, correct Clock/Manager, gradient keys, and layer Environment Tint Weight.
Sun/Moon is not visible	Check assigned sprites/renderer references, Show flags, alpha/daylight fading, path radius/centre, camera view, and sorting.
Rays enabled but invisible	Check whether Lighting is BasicTint. Use Lightweight/URP2D and verify sprite, opacity, computed intensity, owner visibility, and quality cap.
URP2D no active output warning	Check the bridge is enabled/active, Output implements the correct interface, and the generated assembly compiled. The button does not configure the Renderer.
Light2D does not illuminate sprites	Check active 2D Renderer, Sprite Lit material, affected sorting layers, intensity, and range. Replace unlit sample materials.
Wind has no visible effect	Check a nonempty Targets list, assigned visual child, nonzero strength/multiplier/bend, and nonzero Direction X. Layer roots are skipped.
Fog missing or alpha conflict	Check band sprites/materials/alpha/position/sorting. Move bands outside layer roots into a standalone atmosphere group.
Clouds do not fade or move	Check sprite/layer/clock, opacity, velocity, and layer enabled state. Look for another writer of the same tint or transform.
Fireflies missing during the day	Expected with Night Only true. Test Hour 0, a valid clock, emission, and qualityEmission. Retune desired values after Configure.
Preset does not replace the sprite	Preserving existing artwork is intended. The preset sprite is used only when createMissingLayers true creates a missing layer.
Transition fights or jumps	Pause the automatic clock. Avoid cloud, transition, and gameplay scripts writing the same layer.tint. Check unique layer names and valid preset references.
UI callback does not work	Check the correct object and public method on UI Button/UnityEvent, Dynamic float for OnTimeChanged, and the sample EventSystem/Input dependencies.
Changes disappear after stopping play	Unity runtime edits normally do not persist. Author in Edit Mode and save the scene/prefab. Runtime CaptureFrom does not save to disk.

A null-safe call may still produce no output

Check the correct environment reference, active components, sprites/bands/materials, and controller enabled state. The API uses your configuration; it does not automatically create all scene resources.

08 · COMPLETION

Final guided checklist and glossary

Complete your first scene

- 1 In a project without import/compile errors, create an environment and assign an unrotated Orthographic Camera.
- 2 Configure 3-4 uniquely named layers with different factors and sorting relative to the player. Move the camera to inspect depth.
- 3 Enable looping for tileable layers and test motion, zoom, and long travel. Do not assume generated tiles have gameplay colliders.
- 4 Enable Clock and Environment Colours; test Hour 0/6/12/18. Add Sun/Moon visuals, Lightweight rays, and atmosphere one feature at a time as needed.
- 5 Assign fog bands, cloud sprite, wind visual targets, and particle material/style. Empty lists and None fields do not supply artwork.
- 6 Create presets with unique Layer Names and bind the required API scripts/button callbacks. Check transition clock ownership and multiple-effect references.
- 7 Test Low/Medium/High quality, pause controls, preset application, and disable/re-enable. Save scene/prefab settings in Edit Mode.
- 8 Build for the target platform and test actual device performance, rendering, input, and shader behaviour.

Term	Meaning in this package
Manager	The component that owns the environment's layer registry and motion evaluation.
Factor / Baseline	Camera displacement multiplier / reference geometry position.
Drift	An offset added by auto scrolling or clouds, separate from camera displacement.
World units / PPU	Unity scene coordinates / the number of pixels corresponding to one world unit.
Sorting Order	Draw priority within one sorting layer; higher values appear in front.
Preset / Transition	A reusable settings asset / a smooth change from current settings to destination values.
Pool / Tile budget	Reusable generated sprites / the maximum tile cap per scroller.

Check Your scene now covers basic parallax, weather, and runtime controls. Connect the required API calls to your production camera, gameplay, and save systems.

09 · SOURCES

Sources, version scope, and code files

This guide uses the three supplied branding images and 15 Unity screenshots. Image text and field values are reference data, not instructions overriding your request. Sections without a layer-settings screenshot were checked against the current package source.

Package sources

Base folder: `Assets/ERAGANX/2D Parallax/`. Sources read for this guide:

Topic	Source paths (package or repository)
Setup / compatibility	README.md; COMPATIBILITY.md; repository Documentation/ contains INSTALLATION.md, QUICK_START.md, EDITOR_GUIDE.md, SAMPLE_SCENES.md and RUNTIME_API.md.
Editor controls	Editor/Menus/ParallaxMenus.cs; Editor/Inspectors/ParallaxInspector.cs; Editor/Windows/ParallaxControlPanel.cs.
Core API	Parallax2D components in Runtime/Core, Camera, Layers, Scrolling, and Environment, plus ParallaxMath.cs.
Effects / presets	Controller, preset, and transition implementations in Runtime/Lighting, Effects, and API.
URP / samples	Integrations/URP/Parallax2DURPLightBridge.cs; Samples/Shared/ParallaxDemoController.cs and ParallaxDemoPlayer.cs.

Official Unity references

- [Unity 6.3 - Sprite texture import settings](#)
- [Unity 6.3 - 2D Renderer Data](#)
- [Unity 6.3 - Prepare sprites for 2D lighting](#)

Verified scope

Source target: Unity 6000.3.13f1; connected project: URP 17.3.0. This guide describes the development source revision. It covers orthographic/world XY motion and axis-aligned Simple SpriteRenderer looping. It does not guarantee HDRP, perspective/rotated cameras, Tilemap auto-looping, per-tile gameplay cloning, or URP 18+. Player builds and device tests were not run while preparing the guide.

The full C# examples are in `Samples/API Examples/`. API signatures were checked against the current package code. GuideCameraMove and GuideCameraSource are basic helpers; other scripts use Inspector references. Attach only the example components you need and assign their Inspector references.