Easy Exponential Height Fog — URP

Overview

Thank you for Downloading the Easy Exponential Height Fog URP!
This fog system is a physically-based, fully runtime-controllable height fog asset for URP. It uses analytic exponential density along the true view ray for accurate horizon fog, altitude falloff, and sky gradients.

Support

The fastest route for support is currently the form here: https://zdev-studios.com/contact or through the Unity Asset Store portal

---

Table of Contents

Requirements

- Unity **2022.3 LTS** or newer
- **Universal Render Pipeline (URP)** 14+
- URP renderer with **Depth Texture** enabled (`Renderer Asset → Depth Texture: Enabled`)
- RenderGraph enabled (default in URP 14+)

Setup

1. Add the Renderer Feature

The fog is implemented as a URP `ScriptableRendererFeature` . It must be added to your URP Renderer asset before it will render.

Option A — via the Controller Inspector (recommended)

1. Create a fog controller: **GameObject → Effects → Exponential Height Fog Controller**
2. Select the spawned GameObject.
3. In the Inspector, under **Renderer Feature**, click **Find Feature in Renderer**.
    - If the feature already exists in your renderer, it will be assigned automatically.
4. If nothing is found, click **Add Feature to Renderer**.

- This creates the feature as a sub-asset of your renderer data and assigns it immediately. No domain reload is required.

Option B — manually via the Renderer asset

1. Select your URP Renderer asset (`Project → Assets → Settings → [YourRenderer].asset`).
2. Scroll to **Renderer Features** and click **+**.
3. Select **Exponential Height Fog**.

---

2. Enable Depth Texture

The fullscreen fog pass reads `_CameraDepthTexture` to reconstruct world positions.

1. Select your **URP Renderer asset**.
2. Ensure **Depth Texture** is set to **Enabled** (not *Camera*).

If depth is not available, the feature logs a warning and skips the pass silently.

---

3. Place the Controller

The `ExponentialHeightFogController` singleton pushes fog settings to the renderer feature every frame, enabling full runtime and animation control.

- Only one controller should exist per scene. Duplicates are automatically destroyed.
- The controller persists across scene loads and re-syncs settings on each `sceneLoaded` event.
- All public fields can be driven by **Timeline**, **Animation clips**, or **script**.

---

4. Runtime Control via Script

```
// Get the singleton
var fog = ExponentialHeightFogController.Instance;

// Change settings directly — they are pushed to the renderer every Update
fog.fogDensity    = 0.08f;
fog.fogColor      = Color.grey;
fog.heightFalloff = 0.2f;

// Or pull the current feature values into the controller
fog.PullFromFeature();
```

Controller Properties

All properties below are serialized public fields on `ExponentialHeightFogController` . Changes take effect the next frame.

Fog

Property	Type	Default	Description
`fogColor`	HDR Color	`(0.5, 0.6, 0.7, 1)`	Fog color. Alpha multiplies the final fog opacity. Supports HDR.
`fogDensity`	`float` 0-1	`0.05`	Overall volumetric density of the fog (ρ in the analytic formula).

Property	Type	Default	Description
`heightFalloff`	`float` 0–1	`0.1`	Rate at which fog thins with altitude (k). Higher values create a sharper fog floor.
`fogBaseHeight`	`float`	`0`	World-space Y coordinate at which fog is densest. Fog falls off exponentially above this.
`maxFogAmount`	`float` 0–1	`0.9`	Hard maximum opacity the fog can reach. Prevents fully opaque fog even at high density.

Distance

Property	Type	Default	Description
`fogStart`	`float` ≥ 0	`0`	Distance from the camera at which fog begins to fade in.
`fogEnd`	`float` ≥ 0	`500`	Distance at which the fog distance ramp reaches full strength.
`fogFarClip`	`float` ≥ 0	`2000`	Hard distance cutoff — geometry fog is completely suppressed beyond this distance. Sky fog is unaffected.

> **Note:** `fogEnd` controls where the *distance ramp* completes, not a visibility limit. Fog can still be thick at short distances due to height falloff. `fogFarClip` is the true hard cutoff.

Environment Lighting

Property	Type	Default	Description
`useEnvironmentColor`	`bool`	`false`	Tints the fog color using the scene's ambient sky color (`RenderSettings.ambientSkyColor`). Useful for day/night cycles.
`environmentColorInfluence`	`float` 0–1	`0.5`	Blend weight between `fogColor` (0) and the ambient-tinted color (1).

Noise

Property	Type	Default	Description
`useNoise`	`bool`	`false`	Enables a scrolling world-space noise texture to break up the fog uniformly. Requires a noise texture to be assigned.
`noiseTexture`	`Texture2D`	`null`	Greyscale texture. The R channel is sampled. Tileable noise textures work best.
`noiseScrollSpeed`	`Vector2`	`(0.01, 0.005)`	World-space scroll velocity of the noise on the X and Z axes (units/second).
`noiseTiling`	`float` ≥ 0.1	`10`	World-space tiling scale of the noise. Smaller values = larger noise features.
`noiseStrength`	`float` 0–1	`0.5`	How strongly the noise modulates fog density. `0` = no effect, `1` = full modulation range.

Sky Noise

Controls the noise applied specifically to skybox fog. Independent from geometry noise so the sky can have a different scale/movement to feel more like distant atmospheric haze.

Property	Type	Default	Description
`skyNoiseScrollSpeed`	`Vector2`	`(0.005, 0.002)`	Scroll speed for sky noise (view-direction XZ space).
`skyNoiseTiling`	`float` ≥ 0.1	`50`	Tiling scale for sky noise. Higher values = finer noise on the sky dome.
`skyNoiseStrength`	`float` 0–1	`0.3`	Modulation strength for sky noise.

Custom Shader Integration

Transparent and alpha-cutout shaders cannot use the fullscreen fog pass reliably — the pass reads `_CameraDepthTexture` , which may contain background depth at their pixels rather than their own surface depth. The correct approach is to compute height fog inline from the shader's own world position.

Step 1 — Include the library

After your `Core.hlsl` include, add:

```
#include "Assets/ZDev-Assets/ExponentialHeightFog/Runtime/ExponentialHeightFogLib.hlsl"
```

This declares all required globals (`_FogColor` , `_FogDensity` , `_HeightFalloff` , etc.) and provides two functions:

```
// Geometry fog — use for surfaces with a known world position
float ComputeExponentialHeightFog(float3 camPos, float3 worldPos);

// Sky fog — use for background/skybox rays (no surface hit)
float ComputeExponentialHeightFogSky(float3 camPos, float3 rayDir);
```

> **Important:** Declare these outside any `CBUFFER_START(UnityPerMaterial)` block. They are global uniforms set via `Shader.SetGlobal*` and must live in the global constant buffer to be SRP Batcher compatible.

Step 2 — Apply fog in the fragment shader

```
// In your fragment shader, after computing your final lit color:
float  fogFactor = ComputeExponentialHeightFog(GetCameraPositionWS(), worldPos);
color.rgb = lerp(color.rgb, _FogColor.rgb, fogFactor * _FogColor.a);
```

Step 3 — Write the stencil exclusion bit

The fullscreen fog pass skips pixels with **stencil bit 7 (value 128)** set. If your shader applies inline fog, you must write this bit to prevent the fullscreen pass from fogging those pixels a second time.

Add this `Stencil` block inside your `Pass` :

```
Stencil
{
    Ref      128
    WriteMask 128
    Comp     Always
    Pass     Replace
}
```

`WriteMask 128` isolates bit 7 only — all other stencil bits used by URP (decals, light layers, etc.) are unaffected.

Complete minimal example

```
Pass
{
    Stencil
    {
        Ref      128
        WriteMask 128
        Comp     Always
        Pass     Replace
    }

    HLSLPROGRAM
```

```
        #pragma vertex vert
        #pragma fragment frag

        #include "Packages/com.unity.render-pipelines.universal/ShaderLibrary/Core.hlsl"
        #include "Assets/ZDev-Assets/ExponentialHeightFog/Runtime/ExponentialHeightFogLib.hlsl

        // ... your structs, CBUFFER, vertex shader ...

        half4 frag(Varyings IN) : SV_Target
        {
            half3 color = /* your lighting result */;

            float fog = ComputeExponentialHeightFog(GetCameraPositionWS(), IN.positionWS);
            color = lerp(color, _FogColor.rgb, fog * _FogColor.a);

            return half4(color, 1.0);
        }
        ENDHLSL
    }
```

Stencil bit reservation

Bit	Value	Reserved by
7	128	Exponential Height Fog — inline fog exclusion

If another asset in your project uses stencil bit 7, change the `Ref` and `WriteMask` values to an unused bit in both `ExponentialHeightFog.shader` and your custom shaders, keeping them consistent.

Troubleshooting

The demo scene is pink

The most likely issue is that the project is not setup to use the Universal Render Pipeline. This asset requires at the very least URP 14+ and will not work in Built-In RP or HDRP. It is recommended to follow the official URP installation guide: https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@7.1/manual/InstallURPIntoAProject.html

---

My shader doesn't receive any fog

1. Is the controller active?
Check that an `ExponentialHeightFogController` exists in the scene and has a valid **Fog Feature** assigned. Without it, `Shader.SetGlobal*` is never called and all fog uniforms default to zero.

2. Is the fog density above zero?
`fogDensity = 0` produces no fog regardless of other settings. Start with `0.05` and increase.

3. Are you including the library?
Your shader must have `#include "Assets/ZDev-Assets/ExponentialHeightFog/Runtime/ExponentialHeightFogLib.hlsl"` to declare the globals and access `ComputeExponentialHeightFog`. Without it, the uniforms are undeclared and the compiler will error or silently use zero.

4. Are the globals declared inside a CBUFFER?
Global uniforms set by `Shader.SetGlobal*` must **not** be inside a `CBUFFER_START / CBUFFER_END` block. If they are inside `UnityPerMaterial`, the SRP Batcher will override them with per-material values (which default to zero).

The fog looks correct on opaque geometry but not on my transparent shader

The fullscreen fog pass reads `_CameraDepthTexture`, which is captured before transparent objects render. For transparent shaders, follow the Custom Shader Integration guide to compute fog inline and write the stencil exclusion bit.

The fog appears behind my transparent/cutout object instead of on it

This is the stencil exclusion bit missing. The fullscreen pass is reading the background depth at your object's screen pixels and computing fog for the geometry *behind* it.

Fix: add the stencil write block to your pass (see Step 3).

Fog appears on top of my water's underwater scene

The water shader uses `SampleSceneColor` to render what's beneath the surface. That color is captured from the opaque pass before the fog pass runs, so it should already be fog-free. If it isn't:

- Ensure the water shader is computing inline fog from `IN.positionWS` (the surface), not from the underwater depth.
- Ensure the water pass writes stencil bit 128 so the fullscreen pass skips it.

Very distant geometry (500+ units) is suddenly treated as sky

In reversed-Z rendering (the default in URP), geometry at the far clip plane has raw depth values very close to `0.0`. The fog pass uses `Linear01Depth` to detect sky pixels (`>= 0.9999`) rather than a raw depth threshold, which is platform-independent. If you are seeing this:

- Increase your camera's far clip plane — geometry beyond it is clipped and will read as sky depth.
- Check `fogFarClip` on the controller. If it is lower than the distance to that geometry, fog is being zeroed out and the sky fog bleeds through.

The fog doesn't appear until I reload the domain / restart Play mode

The controller calls `fogFeature.Create()` in `Awake` when `IsMaterialMissing` is true, which forces the pass and its material to initialize immediately. If this doesn't work:

- Use the **Find Feature in Renderer** or **Add Feature to Renderer** buttons in the controller Inspector — these call `SetFeatureAndInitialize()` which bypasses the serialization delay.
- Ensure the renderer feature is saved as a sub-asset of the renderer data asset, not as a loose `ScriptableObject` in the scene.

The fog doesn't react to my day/night cycle

Enable **Use Environment Color** on the controller and set **Environment Color Influence** above zero. The fog color is then lerped toward `RenderSettings.ambientSkyColor` each frame, which Unity updates automatically when you rotate the directional light or use a sky material with dynamic exposure.

Noise has no effect

- Ensure **Use Noise** is enabled and a texture is assigned to **Noise Texture**.
- The noise texture must be **Read/Write Enabled** in its import settings if you modify it at runtime.
- Check that **Noise Strength** is above zero.
- If the noise looks frozen, increase **Noise Scroll Speed**.

For additional support, refer to the inline HLSL comments in `ExponentialHeightFogLib.hlsl` and `ExponentialHeightFog.shader`.