AE AREOPAGUS ENGINEERING

Cinematic Car Paint

For the love of looking good, fast.



v1.0 — August 2026

USER MANUAL

Introduction

Cinematic Car Paint contains four systems:

- A layered, physically based automotive paint shader with base coat, metallic flakes, clear coat, optional pearl and iridescence, procedural decay and under-clearcoat decals.
- A custom hardware ray-traced reflection system for URP.
- A GPU rain simulation for painted bodywork and glass.
- An editor rust-growth simulation that can be baked for runtime ageing.

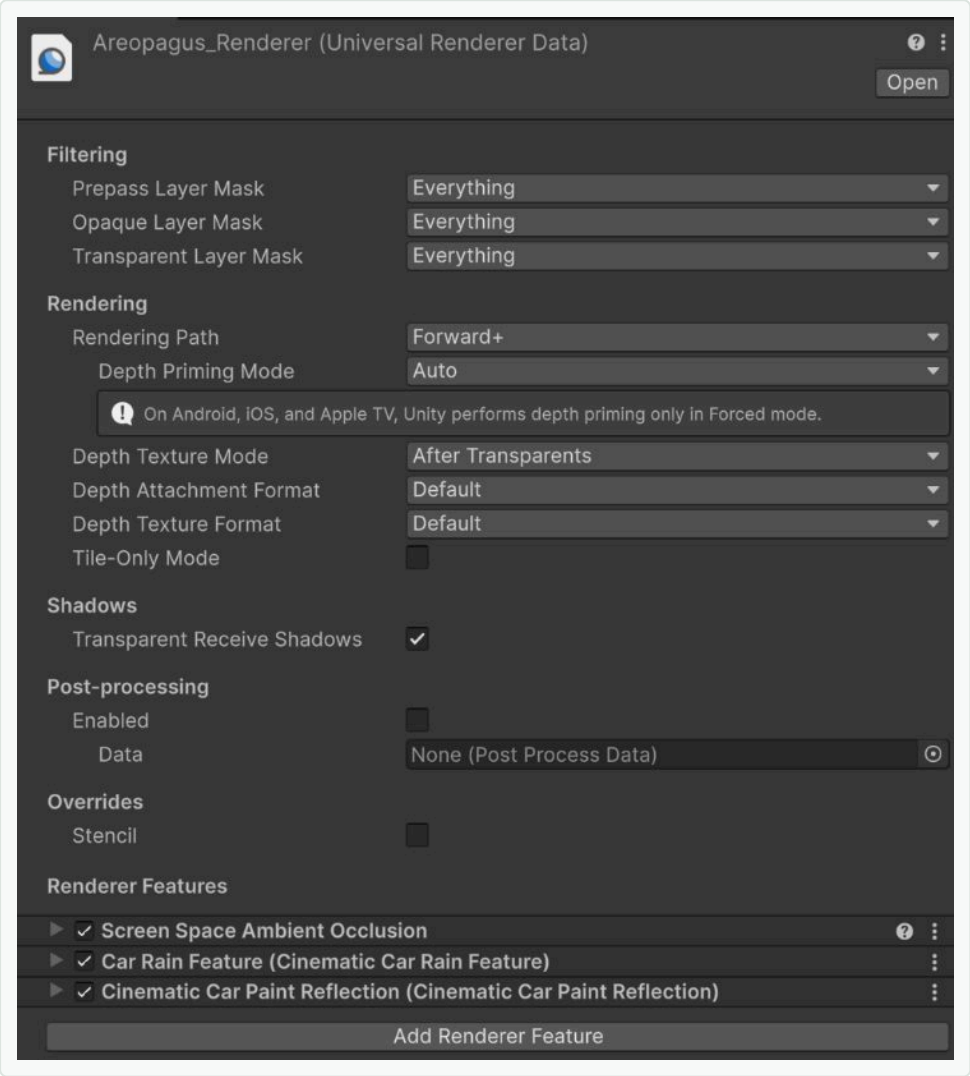
Only the paint material is required. The rain, reflection or simulated-rust features are optional (add them when needed)



> **Note:** These are extensive and complex shaders; incorporating as many features as we could possibly pack into them. When importing, or creating new materials, be patient, they will require some time to compile.

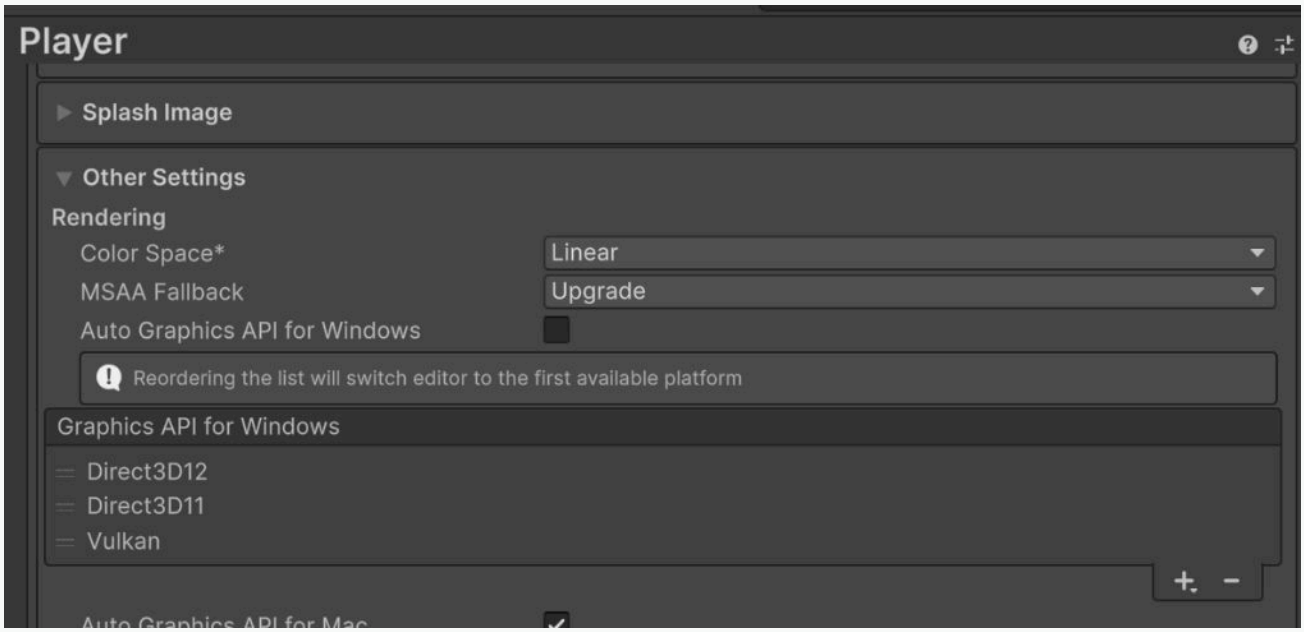
Quick Start

1. Import the package and add the two featured renderer features to your Universal Renderer (or alternatively, make use of the Areopagus renderer settings provided in 'Settings/'). These are the *Cinematic Car Paint Reflection* feature and the *Cinematic Car Rain* feature.



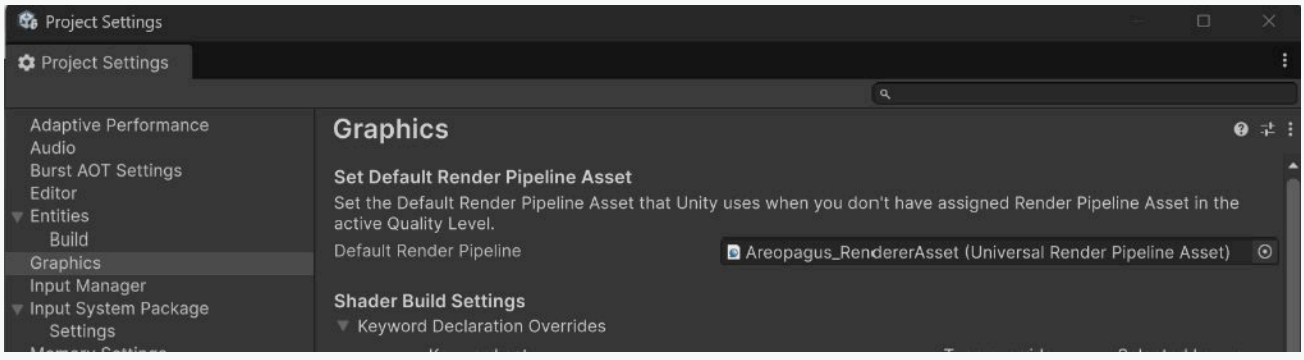
Example of the Cinematic Car Paint render features on the included URP renderer

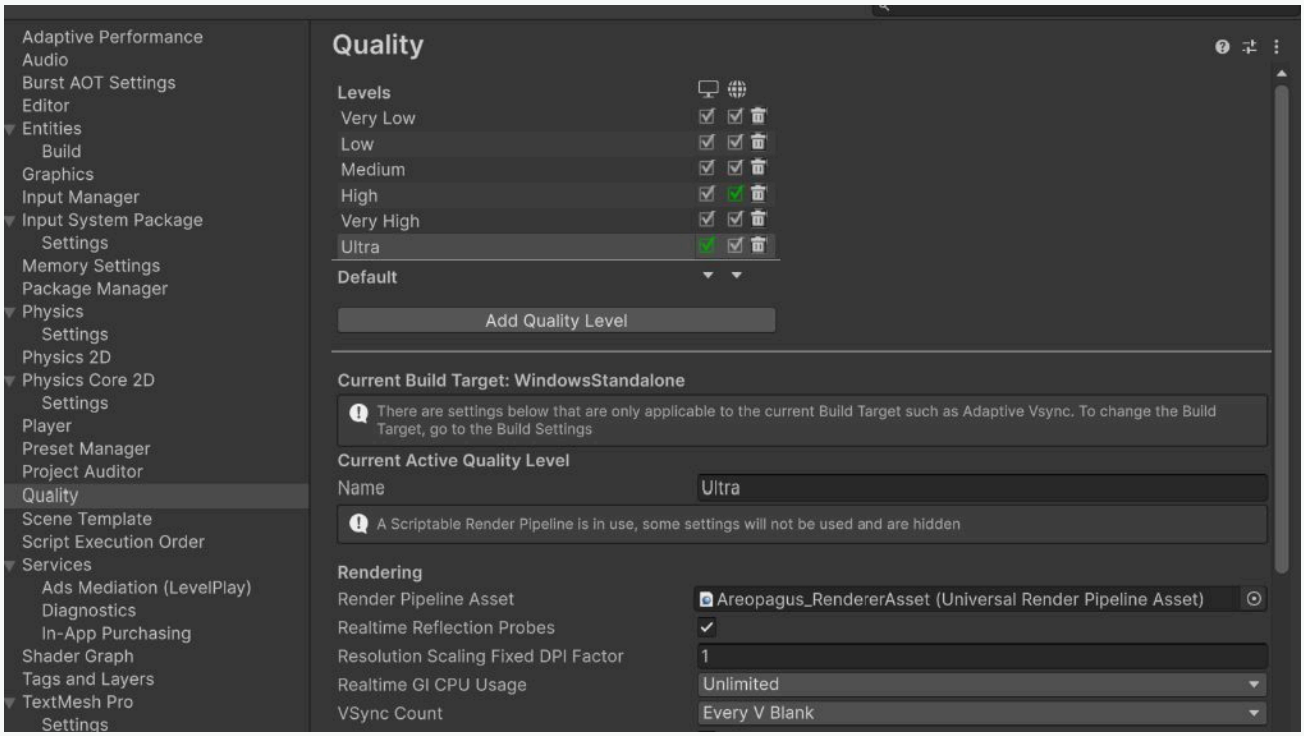
Also ensure you set the graphics API to Direct3D12 if you want to make use of the ray-traced reflections (and ensure its the first in line) - they will not work in D3D11 or Vulkan. But all other features (rain, rust, normal car paint etc.) will.



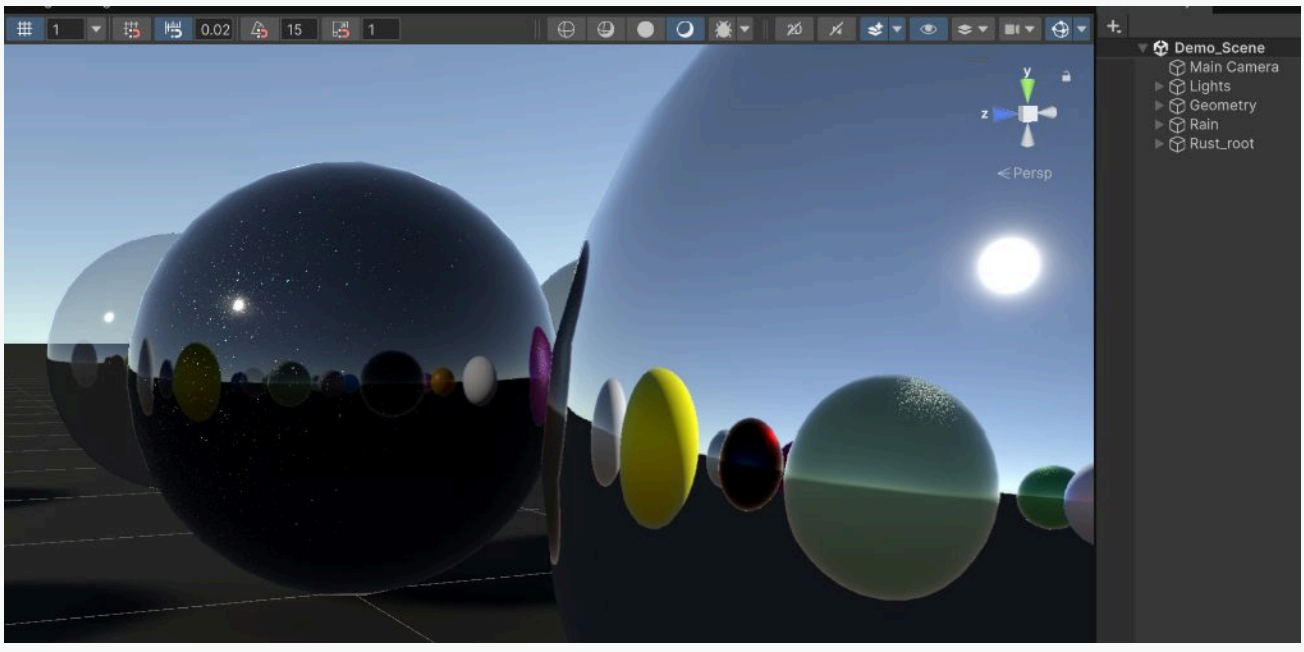
Set your Graphics APIs (D3D12 for raytraced reflections)

> **Note:** If you are utilising the included renderer asset in the package, then ensure the correct default render pipeline and pipeline asset is set in the Graphics and Quality settings.





2. Open `Demo_Scene.unity` to check that the shaders and materials imported correctly. It will likely take some time to compile all the shaders. Be patient! :)



3. Duplicate a ready-made material from `Materials/` into your own project folder, or create a material with the **Areopagus/Cinematic Car Paint** shader, and assign it to the vehicle body.

4. Note that ANY component requires a Cinematic Car Paint Vehicle component at the root of the vehicle (or object). Otherwise the shaders will not function correctly and you'll get 'see through' surfaces. Optionally:- add a Cinematic Car Paint Reflection Probe (not a standard Unity reflection probe component) - for local reflections specific to the car.
5. For rain, object-space decals or simulated rust, choose **Add Component → Areopagus → Cinematic Car Paint Vehicle** on the vehicle root. Use the paint material on each registered body panel.
6. For rain, add **Cinematic Car Rain Feature** to the active Universal Renderer Data asset, and enable **Opaque Texture** on the URP asset.
7. For custom DXR reflections, add **Cinematic Car Paint Reflection** to the same renderer data asset. On each rigid MeshRenderer that should appear in reflections, set **Ray Tracing Mode** to a value other than **Off**.

> Note: The scene needs lighting and either an environment reflection source or a reflection probe. The material works without either renderer feature, and reflection probes remain its reflection source whenever the custom DXR path is unavailable.

Package contents

- `Materials/`: supplied car-paint and glass-rain materials.
- `Shaders/`: paint, glass, rain, rust, decal and ray-tracing shaders.
- `Scripts/`: runtime vehicle, rain, reflection, reflection-probe and rust-bake components.
- `Editor/`: custom inspectors, setup diagnostics and rust-simulation tools.
- `Settings/`: a URP asset and Universal Renderer Data with the rain and reflection renderer features added.
- `Demo_Scene.unity` and `Demo_Scene/`: the Unity-primitive demonstration scene and its lighting data.
- `BakedAssets/`: the baked rust growth history used by the demo scene.
- `Third-Party Notices.txt`: third-party attribution and licence text.
- `Manual.pdf`: this thing you're reading right here.

The package does not include vehicle models, environment models, driving controllers or a general weather system.

Making a paint material

The supplied materials cover clean factory colours, metallic, candy, pearl, iridescent and weathered finishes. The inspector groups controls by feature, and the less obvious controls carry tooltips. Most paint feature toggles are shader keywords, so a disabled section costs nothing. **Enable Second Decal**, **Use Decay Bias Mask**, **Wipe Probe Reflection (RT only)** and the Glass Rain toggles are runtime switches.

The paint inspector's foldouts are:

- **Base Paint** for the body colour, base response and coat depth.
- **Metallic Flake** for pigment type, size, density and sparkle.
- **Clear Coat** for the outer glossy layer and the reflection strength controls.
- **Artist Overlay** for an optional tinted texture overlay.
- **Colour-Shift FX** for the optional pearl subcoat and iridescence (angle-dependent colour).
- **Procedural Rust / Decay** for surface ageing.
- **Rain** for rain streaks on the body.
- **Decals** for stripes, numbers and wraps beneath the clear coat.
- **Advanced / Diagnostics** for the Projection Preview views and rendering options.

Pearl and iridescence take effect as soon as they are enabled. If a new material changes strongly at grazing angles, adjust **Colour-Shift FX** first.

Custom DXR reflections and probe fallback

The pièce de résistance. The reflection renderer feature is a custom URP pass built on Unity's documented ray-tracing shader and acceleration-structure APIs. It reflects geometry outside the camera view and supports a configurable number of bounces.

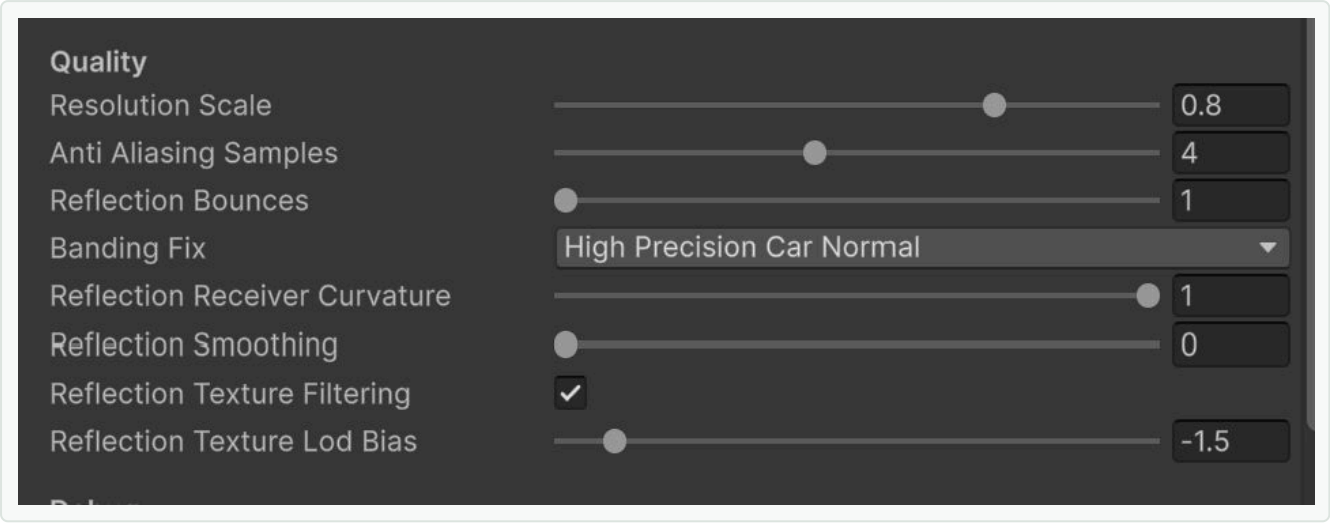


It requires:

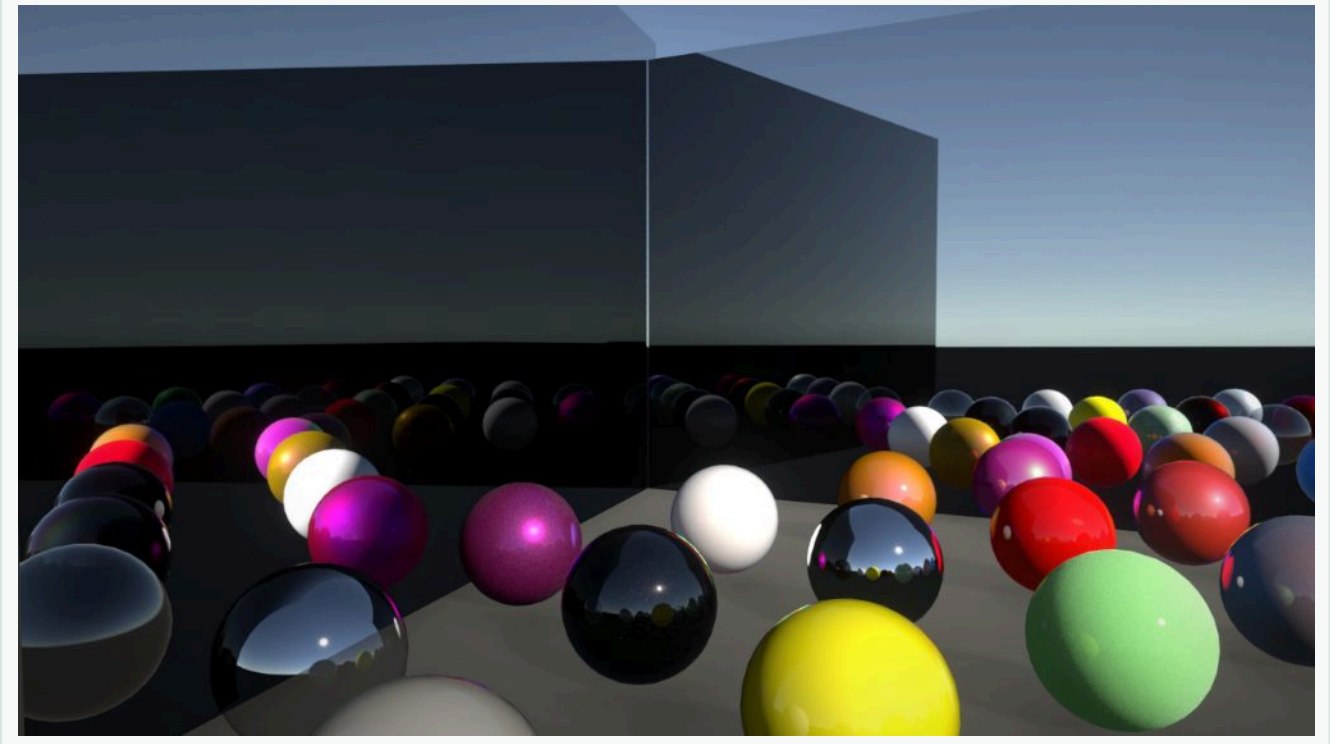
- Direct3D 12 as the active graphics API (Windows only)
- A DXR-capable GPU and driver. RTX-class hardware is recommended; GeForce GTX 10-series cards tested as capable of running, however RTX would provide better performance.

To select Direct3D 12, open **Project Settings** → **Player** → **Other Settings**, disable **Auto Graphics API for Windows**, and place **Direct3D12** first. Test the feature on the hardware range you intend to support. When supporting mixed PC hardware, offer a player quality option for it.

> **Note:** Tune the settings for the quality or performance you need. Cinematic Car Paint supports up to full-res reflections, and 4x multi-bounce rays (i.e. to get a mirror of a mirror)



Example quality tuning settings on the renderer feature



Example of multi-bounce rays. Tune depending on your target hardware

Without compatible hardware or Direct3D 12, the renderer feature does not dispatch and the paint uses reflection probes. For a vehicle-local fallback, choose **Add Component** → **Areopagus** → **Cinematic Car Paint** → **Reflection Probe** on the vehicle. It manages a box-projected probe that follows the vehicle.

> **Note:** This has been tested on a Nvidia 1060 3gb laptop discrete GPU and operates with

> software raytracing quite well. However, RTX or AMD's recent cards are recommended. Screenshots throughout captured on a 9070XT.

Two material controls under **Clear Coat** set the blend:

- **RT Reflection Strength (per material)** scales the custom DXR result.
- **Wipe Probe Reflection (RT only)** removes the probe reflection wherever a valid DXR result exists, so the two sources are not added together.

Reflected LOD Level on the reflection feature (default **LOD 2**) chooses which level of each LODGroup appears in reflections. A group with fewer levels uses its coarsest level, a billboard-only level falls back to the nearest finer level with a mesh, and objects without a LODGroup are unaffected. Vehicles with a **Cinematic Car Paint Vehicle** component are always reflected at full detail (LOD 0). Higher levels put fewer triangles in the reflection structure; terrain trees follow **Tree Reflection Detail** instead.

Enable Periodic Scene Rescan on the reflection feature is off by default. Leave it off unless runtime systems spawn, destroy, enable or disable reflected geometry without telling the feature; scene loads and unloads are detected automatically. When on, every **Scene Rescan Interval Frames** (minimum 30) it re-collects the scene's LOD groups and MeshRenderers and rebuilds the CPU scene index, spread across frames at **Cache Renderers Per Frame**; only streamed tiles whose contents changed are reloaded. Scripts that know when geometry changes should leave the option off and call `CinematicCarPaintReflection.RequestSceneRebuild()` once after each batch of changes.

> **Note:** Tip! If in doubt, check the tooltip. I've given almost every setting a tooltip popup trying to explain what it does and why its there and exposed as many tunable parameters as possible for users.

Rain across a vehicle

For body rain:

1. Add **Cinematic Car Paint Vehicle** to the vehicle root and set **Enable Rain Simulation** on.
2. Add **Cinematic Car Rain Feature** to the active renderer data (if not already done)
3. On the body material, turn on **Enable Rain Streaks** and raise **Rainfall (mm/hr)**.
4. Use **Areopagus/Cinematic Car Glass Rain** for windows and tune its drop controls to match the body.

The vehicle component supplies the object-space frame and motion used by the simulation.



One vehicle runs the full surface-water simulation at a time: the nearest eligible registered vehicle to the active camera, selected automatically. Other registered vehicles stay dry rather than sampling a water field that does not match their body. Not tested in a multi-active-camera scene - be aware.

Procedural and simulated rust

Both workflows require **Enable Procedural Decay** on the paint material.

For procedural rust, raise **Rust Coverage (%)** and tune the decay controls on the material. The material can favour sills, corners and the lower body automatically, and an optional mask can guide placement.

For a vehicle-specific growth history:

1. Add **Cinematic Car Paint Vehicle** to the body root.
2. In **Rust Creation Simulation**, enable **Use Simulated Growth History**.
3. Set the creation controls and click **Simulate Rust Growth**. The editor simulation grows rust over the registered painted panels and can cross nearby panel gaps.
4. Scrub **Rusted Area Fraction** to preview the completed history without running the simulation again.

5. Click **Bake Growth History Texture3D For Runtime...** and save the generated asset inside your project.

The runtime samples the baked history; changing **Rusted Area Fraction** does not re-run the editor solve. Re-simulate after changing registered geometry, the vehicle frame, simulation resolution or creation settings.



The bake stores four 16-bit UNorm history channels (8 bytes per voxel). The arrival channels hold a continuous distance field. The editor solve writes 16-bit UNorm when the GPU supports it, and the preview then matches the bake exactly; otherwise it uses 32-bit float, then 16-bit float, which can show faint banding. A history that does not match the vehicle frame, resolution or simulation controls is not used: the Inspector reports the mismatch and the shader falls back to procedural rust.

The bake replaces the growth simulation at runtime, and the relief intersection search reuses its broad attack/noise channel. Sub-millimetre rust is computed live rather than baked, so it stays sharp without a vehicle-scale 3D texture dense enough to store millimetre voxels. Fine spectra and relief are selected by screen footprint: close-ups keep the full grain stack, and bands smaller than a pixel fade continuously into roughness and stop being evaluated as the camera moves away. **Regional Severity Variation (%)** uses the simulated local age and attack capacity to mix thin young oxide, mature layered scale and deeper severe areas within the same rust coverage.

Decals and stripes

The Decals section places up to two graphics beneath the clear coat. For each object-space placement, choose a **Vehicle Side / Face**, centre, size and rotation.

Right Side and **Left Side** are the vehicle's own sides in the root's object space; you do not need to work out which axis that is. Each is a directional planar projector cast inward from that side. It uses vehicle-forward as the decal's horizontal direction and vehicle-up as its vertical direction, rejects the opposite half of the body, and fades surfaces that turn away from the cast. A door decal therefore stays on its own side and still crosses shallow panel curves. In **Auto** placement, **Mirror Across Centre (twin stripes / both sides)** produces symmetrical side placements, or twin racing stripes on the other projector faces.



Decal 2 has its own textures and placement and shares Decal 1's metallic, AO, smoothness, emissive and clear-coat response controls. Each decal's normal and mask maps are sampled with the **Wrap Mode**, **Filter Mode** and **Aniso Level** of that decal's colour texture (its **Albedo** slot), so set those import settings on the colour texture.

Platform notes and known limitations

- Requires Unity 6.5 (6000.5) or newer with URP 17.5 or newer. Earlier Unity 6 releases are not supported or tested, and assets serialised by Unity 6.5 may not load correctly in them. Earlier

versions of Unity may work (e.g. it has been tested in 6.4 and 6), however these are not supported.

- Tested on Unity **6000.5.6f1** with URP and RenderGraph.
- Forward, Forward+, Deferred and Deferred+ rendering paths are supported.
- Rain and editor rust simulation use compute-shaders. Rain also needs Opaque Texture support, which the Glass Rain shader samples for transmission even when rain is off.
- **Reflected scene geometry is limited to rigid MeshRenderers**. Skinned meshes do not appear in the custom DXR result.
- The custom DXR path does not reflect procedural rust, because of the compute cost.
- Avoid negative scale on reflected geometry.
- Meshes over **Max Mesh Vertices** (default 500,000) are reflected as a simplified proxy when **Simplify Oversized Meshes** is on (the default) and the mesh has Read/Write enabled on import; otherwise they are skipped. **Explain Reflection For Selection** reports this per object. Submeshes that are not triangle lists are skipped without a message.
- Materials the reflection scraper does not recognise use a neutral approximation rather than their full custom shader appearance. The use of **Reflection Overlays** in the renderer feature is experimental.
- The first shader import and compile take a long time: the paint has several passes, and its optional features produce many variants. The included shader stripper removes the Projection Preview debug variants from player builds; it does not shorten the first import.

Troubleshooting

- **Black probe-atlas patches on the paint:** this was a URP 6000.1 bug in testing; update Unity to 6.5+ or, turn off **Probe Blending** on the URP asset
- **An object is absent from custom reflections:** confirm its Ray Tracing Mode is not Off. Select it and run **Tools -> Areopagus -> Cinematic Car Paint -> Explain Reflection For Selection** for an object-specific diagnosis.
- **No custom reflections:** confirm that Direct3D 12 is active and the machine reports ray-tracing shader support. Without them, probe reflections are the intended fallback. In a player build, a `[CinematicCarPaint]` error in the Player.log that names a missing shader means the feature's shader reference was not saved.
- **Reflections look doubled:** enable **Wipe Probe Reflection (RT only)** on the material so the probe and DXR results are not added together.
- **The car is transparent or object-space effects do not align:** confirm that Cinematic Car Paint Vehicle is on the common body root and that its registered renderers cover the intended panels. **Projection Preview (decal/rain placement)** under **Advanced / Diagnostics** shows the

object-space frame and how the registered mesh is being read.

- **Rain is absent:** confirm that the rain renderer feature and Cinematic Car Paint Vehicle are present, **Enable Rain Simulation** and **Enable Rain Streaks** are on, Rainfall is above zero, and the URP asset has **Opaque Texture** enabled. If drops show but never run in a player build, look in the Player.log for a `[CinematicCarPaint]` error that names a missing shader and apply its fix.
- **Baked rust doesn't show in a build:** a painted part of the vehicle is Batching Static. The vehicle Inspector and the build log should notify of this; untick Batching Static on that object. **Do not** mark a vehicle's painted parts **Batching Static**. A build can merge them into a world-space batch, and the vehicle then cannot measure its body. Whereby the baked rust history is not used, and decals and rain are misplaced.
- **The Inspector says the growth history does not match this vehicle frame, resolution, or simulation controls:** the shader is using procedural rust instead. Restore the geometry/settings used for the bake, or run the simulation again (**Simulate Rust Growth** or **Re-simulate Rust Growth**) and create a new runtime bake.

Support

For setup help or technical issues, email **support@areopagus.engineering**. Include the Unity version, URP version, rendering path, active graphics API and the complete Console error or warning, and debug outputs.

Developed by Josh Diyn under the Areopagus Engineering brand.

Copyright © 2026 Josh Diyn. All rights reserved.