Documentation

Raphael Ernaelsten

@RaphErnaelsten

Table Of Contents

About Aura 2

Note : for the sake of clarity and conciseness, this document will possibly refer to Aura 2 as Aura (as a system). We will effectively be talking about Aura 2 and not Aura 1.

Aura is a volumetric lighting (or volumetric fog) solution for Unity.
Aura simulates the scattering of the light in the environmental medium and the illumination of micro-particles that are present in this environment but not big enough to be distinguished by the eye/camera.

The directional light is scattered into the air and illuminates the invisible micro-particles.

Aura uses a globalist approach that makes every lights and injection volumes interconnected. This means that when you locally modify the environment somewhere, it will automatically affect all the lights and injection volumes. No boring and redundant per-instance setup ...
Also, all lights and injection volumes are dynamic and then, can be created, modified or destroyed at runtime.

Aura packs a bunch of features such as full lighting support, injection volumes, light probes, reusable illumination (e.g. for particles) ...

Click here to view the release trailer of Aura 2 (on YouTube).

Aura is not an atmospheric scattering simulation nor a cloud simulator.

Toolbox

Aura 2 comes with an integrated toolbox in the SceneView (the edition window). This toolbox contains useful shortcuts to quickly add and edit the volumetric lighting of your scenes.
It is divided into panels which have different purposes.

SCENE panel

This panel contains shortcuts that will directly add Aura 2 to the existing scene.
The first big button will open the Aura Ambience Presets panel that will apply a general mood to your scene.
The other buttons will add Aura 2 components to the cameras or lights in your scene.

CREATE panel

This panel will give you access to shortcuts that will allow you to rapidly and interactively create Aura Cameras, Aura Lights or Aura Volumes.
Click on a button then click anywhere in the scene to begin the process of creation. Each new click will make a new dimension that you will be able to visually control, until your new object is created.
The two buttons at the bottom of this panel are options for the creation process.
The first one will let you define the referential of the created volume, while the second one will allow Unity to focus and zoom on the new object when it's created.

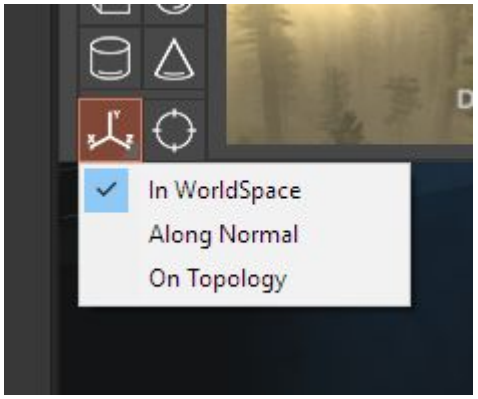
Aura Ambience Presets

The Aura Ambience Presets panels will allow you to choose between a selection of pre-configured ambiences to apply to your scene.

This is the PERFECT KICKSTART for giving a generic mood, that you will later on tweak and refine to fit your vision.

Creation Mode Options

Two options are currently available for the creation mode.

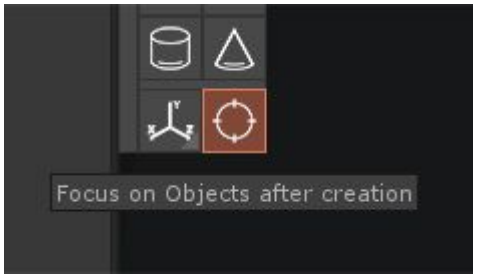


The **Volume Creation Space** option will allow you to define the orientation of the object you will be creating.

In WorldSpace : Will orient the up axis of the object along the Y axis in the world. The resulting object will be then levelled on the XZ plan.

Along Normal : Will orient the up axis of the object along the normal of the object under the mouse. The created object will then orient according to the surface of the object under the mouse.

On Topology : Will orient the object according to the previous click and the position under the mouse. The created object will then be oriented on uneven surfaces or in complex volumes.



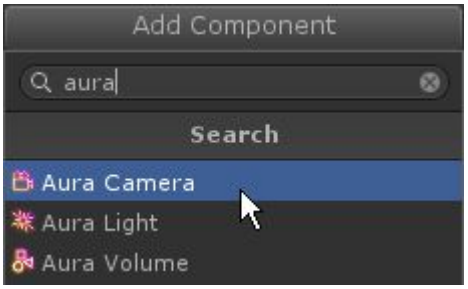
The **Focus On Object After Creation** will allow you to focus and zoom on the newly created Aura Object just after completion of the creation process.

Aura Components

Aura 2 uses GameObject components to be able to start the volumetric lighting/fog computation and feed the data to it. For the sake of clarity, GameObjects with Aura components assigned will be called Aura objects.

Aura objects can be created in several ways :

use the Add Component button at the bottom of the components pile of your GameObject

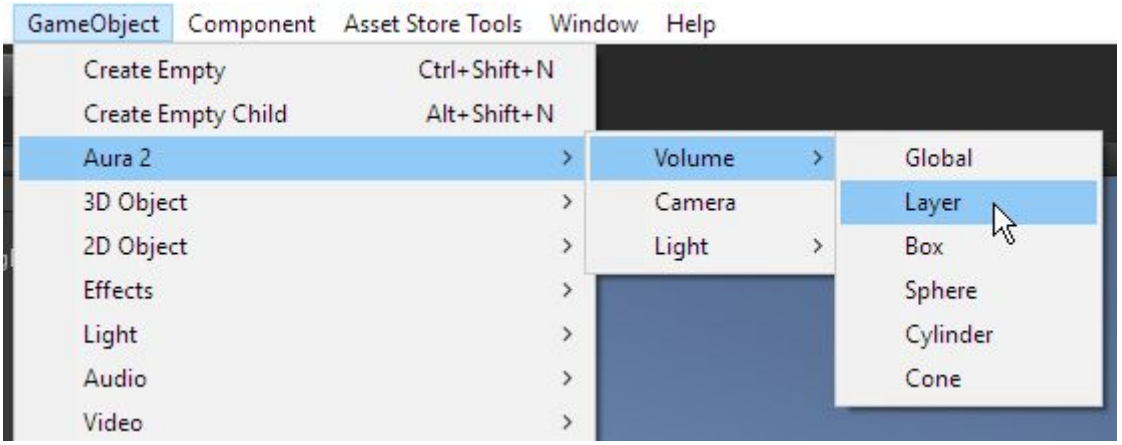


use the Add Aura button on Camera/Light GameObjects

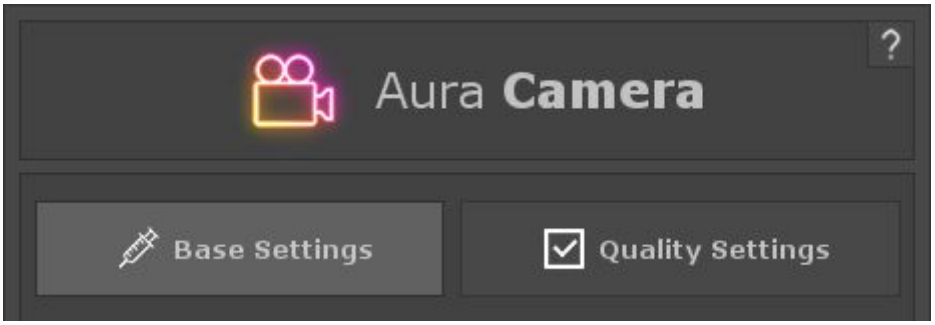
This feature is currently disabled

use the shortcut buttons integrated in the Toolbox

use the Aura 2 menu in the GameObject menu



Aura Camera



Aura Camera components are in charge of collecting all the contributing data, processing the Aura system computation and then displaying the volumetric lighting.

The first function is to establish a foundation for the volumetric lighting, using a **Aura Base Settings** preset file.

The second responsibility is to setup the quality of the volumetric lighting computation, using a **Aura Quality Settings** preset file.

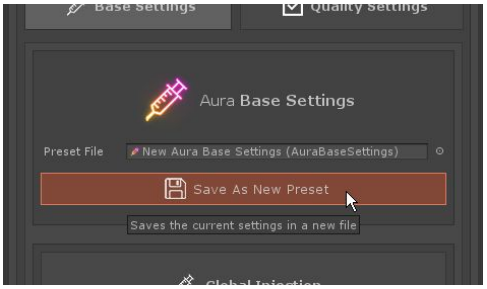
Using those two collections of parameters, the **Aura Camera** component will then gather all the data from the contributing Aura components, process and display the volumetric lighting computation.

Aura Base Settings preset files

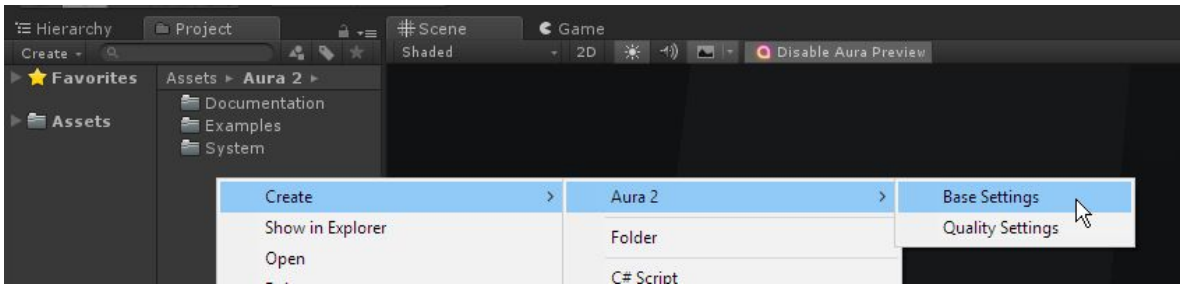
Aura Base Settings preset files are used to set the base parameters of the volumetric lighting such as the global density, scattering, extinction etc…

Since Aura 2 now supports multiple cameras at the same time, you will want to be able to set the same fundamental settings to those cameras. This is the reason why those base settings are now encapsulated in preset files, so you will be able to assign the same preset file to multiple cameras at the same time.

To create a new **Aura Base Settings** preset file you can :

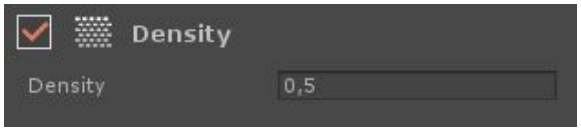


Use the **Save As New Preset** button in the Aura Camera component

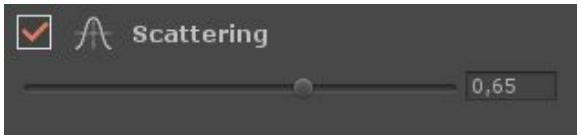


Use the **Create -> Aura 2 -> Base Settings** menu in the Project Window

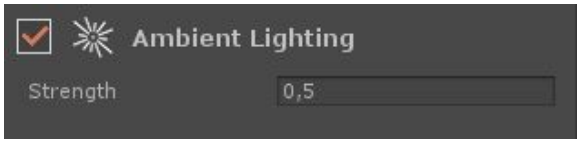
The **Global Injection** section will let you setup the starting parameters for your environment.



The **Density** option will allow you to inject a starting ambient fog density in your scene. It is recommended to use this value as very low, since you'll be able to locally increase the fog density with Aura Volumes.



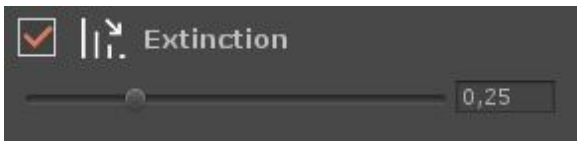
The **Scattering** option will allow you to setup the default scattering factor of the lights in your scene. The scattering factor is how much light will be scattered by the micro-particles before reaching the eye/camera. It will result in more focussed volumetric lights with a low factor or on the contrary, very smooth and soft volumetric lights with a high factor.



The **Ambient Lighting** option will allow you to inject the ambient lighting (from the *Lighting* window) of your scene in the Aura system, resulting in a more unified and integrated volumetric fog. This option is very complete as it will work with flat colors, gradient colors and even skyboxes. It will even take the light scattering into consideration.

The **Tint** option will allow you to colorize the volumetric lighting. You can use this option to tint the basic colorless fog.

The **Light** option will allow you to simply inject a global ambient lighting into the Aura system.



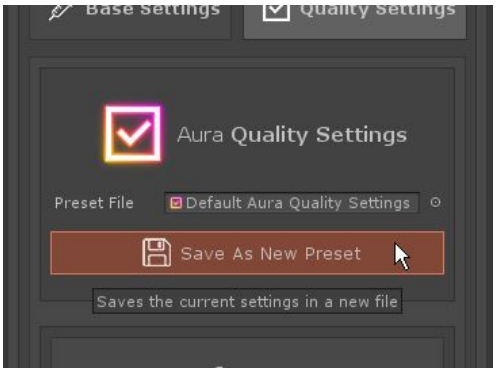
The **Extinction** option will allow you to enable the physical absorption of light along distance. It will gradually decay the volumetric lighting as it gets further from the camera.

Aura Quality Settings preset files

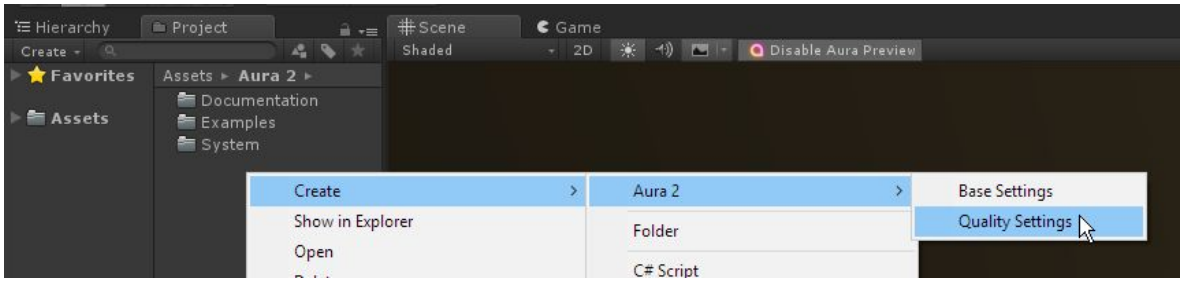
Aura Quality Settings preset files are used to set the quality parameters of the volumetric lighting computation such as the resolution, what feature to reject, the reprojection ...

Since Aura 2 now supports multiple cameras at the same time, you will want to be able to set the same settings to those cameras or to create several quality settings presets and switch between them to switch quality at runtime.
Those are the reasons why quality settings are now encapsulated in preset files.

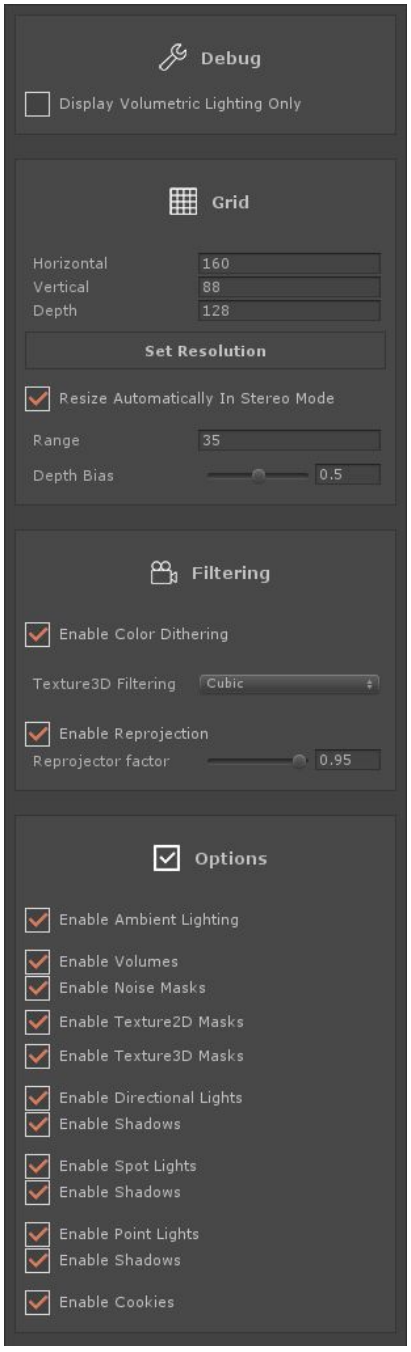
To create a new **Aura Quality Settings** preset file you can :



Use the **Save As New Preset** button in the Aura Camera component



Use the **Create -> Aura 2 -> Quality Settings** menu in the Project Window



The **Display Volumetric Lighting Only** option will allow you to only visualize the volumetric lighting of the scene. This is particularly useful when you want to fine tweaks areas or track unexpected fog results.

The **Grid Resolution** option will let you to define the accuracy of the volumetric computation. The higher the grid is, the finer the volumetric lighting will be, but be careful because the computation cost will consequently increase as well.

The **Resize Automatically In Stereo Mode** option will automatically detect if the camera is in stereo mode and will change the width resolution accordingly to keep the same local quality. This option is useful when you want to keep the same Aura Quality Settings file for mono and stereo mode. *Note regarding VR/Stereo : using a volumetric lighting solution is per se already costy. Combining it with a stereo mode which will need twice the computation of the effect will dramatically increase the computation needs as well.*

The **Range** option will let you define the maximum distance to which the volumetric lighting will be computed. The depth resolution will then be distributed from the camera to this distance.

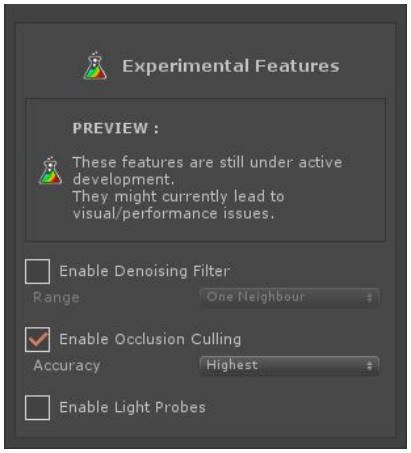
The **Depth Bias** option will allow you modify the distribution of the depth resolution up to the maximum distance. Zero will mean that the distribution is unbiased, the depth slices will be then linearly distributed. The higher the value, the more the slices will be weighted towards the camera. Because the further from the camera, the less accuracy we need.

The **Enable Color Dithering** option will dither the volumetric lighting colors to avoid banding due to limited display precision.

The **Texture3D Filtering** will influence how data are interpolated between the cells of the resolution grid.

The **Reprojection** option will temporally reproject the volumetric lighting frame after frame. This will smoother the volumetric lighting but also induce some small ghosting.

The **Contribution** options will let you choose what feature will be included in the volumetric computation.

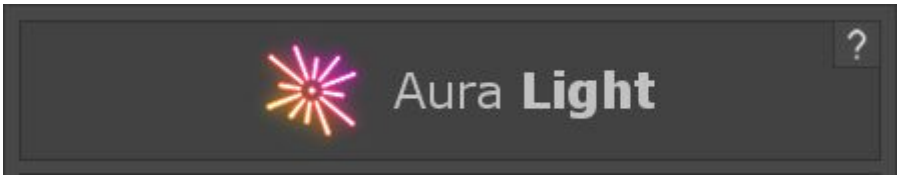


The **Denoising Filter** option will remove noise from the volumetric lighting.

The **Occlusion Culling** option will detect spaces that are hidden to the camera and will exclude the volumetric computation from them.

The **Light Probes** option will enable the light probes lighting injection.

Aura Lights



Aura Light components are in charge of collecting the light's data/parameters and providing them to the Aura system.

The **Aura Light** components can be assigned on the Lights you want to be taken into account into the volumetric lighting computation.

Aura Light components are divided in several panels :

- **Common Parameters** panel
  This panel contains settings that are identical to all types of lights.

Enable Shadows : will allow the volumetric shadows to be computed

Enable Cookies : will allow the volumetric cookies to be computed

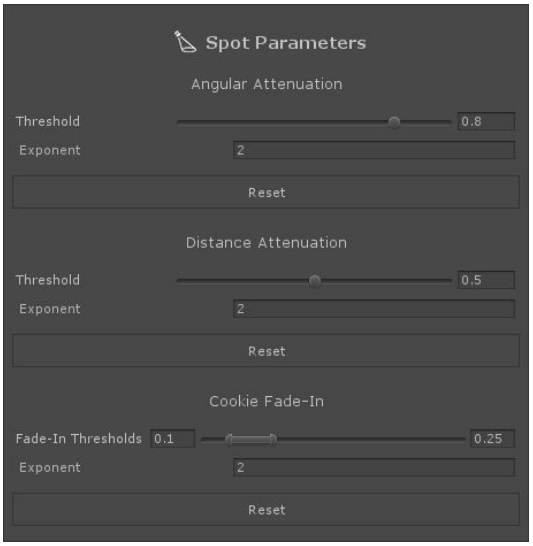
Override Color : will replace the color of the volumetric light

Scattering Bias : will change the scattering of this light only

- **Directional Parameters** panels
  This panel contains settings that are unique to the directional lights.

Enable Out-Of-Phase Color : this option will allow you to interpolate towards a custom color according to the scattering.

- **Spot Parameters** panels
  This panel contains settings that are unique to the spot lights.

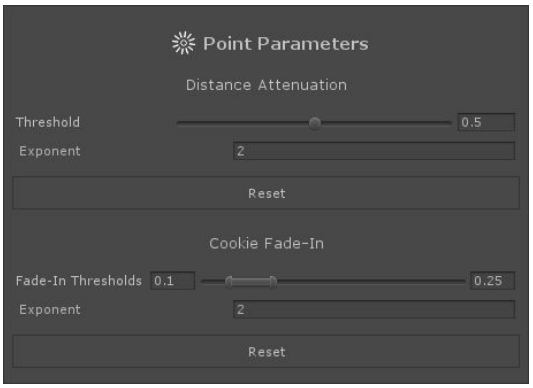


Angular Attenuation : this option will let you tweak the attenuation going from the center of the spot towards the outer angular borders. The **Threshold** is the normalized percentage where the attenuation will start and the **Exponent** is the curve.

Distance Attenuation : this option will let you tweak the attenuation starting from the spot to outer distance borders. The **Threshold** is the normalized percentage where the attenuation will start and the **Exponent** is the curve.

Cookie Fade-In : this option will let you tweak the way the cookie will appear. The **Fade-In Thresholds** are the normalized percentages where the cookie will start appearing and where it will be totally appeared and the **Exponent** is the curve.

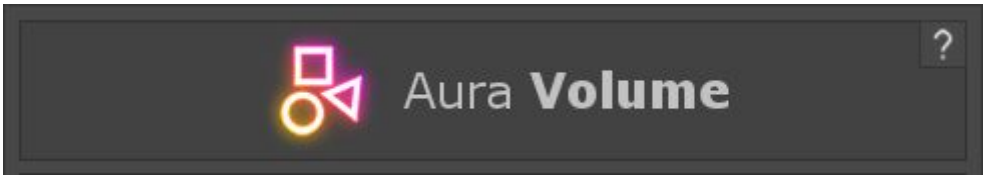
- **Point Parameters** panels
  This panel contains settings that are unique to the point lights.



Distance Attenuation : this option will let you tweak the attenuation starting from the spot to outer distance borders. The **Threshold** is the normalized percentage where the attenuation will start and the **Exponent** is the curve.

Cookie Fade-In : this option will let you tweak the way the cookie will appear. The **Fade-In Thresholds** are the normalized percentages where the cookie will start appearing and where it will be totally appeared and the **Exponent** is the curve.

Aura Volumes



Aura Volume components can be added into the scenes to locally modify one or several of the base ingredients of Aura's system (Density, Color and Scattering).

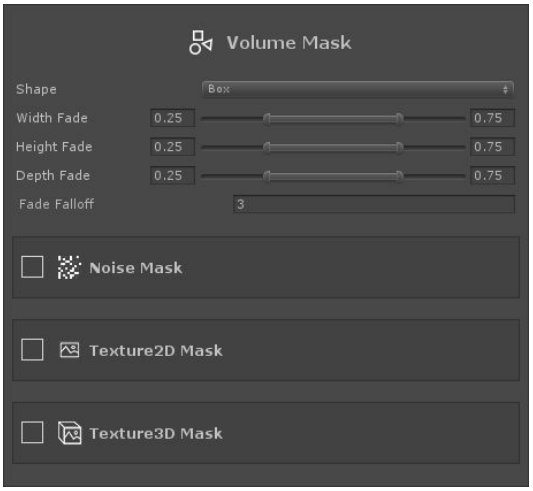
Aura Volume components are in charge of creating an injection volume, collecting its data/parameters and providing them to the Aura system.

Aura Volume objects are responsible of injecting data inside the Aura system.

Three types of data can be injected inside the system: fog **density**, pure **light** and lighting **scattering** factor.

Aura Volume components are divided in two panels :

- **Volume Mask** panel
  This panel contains the parameters used to setup the injection mask. This mask will be used to mask out the **Injected Data**.



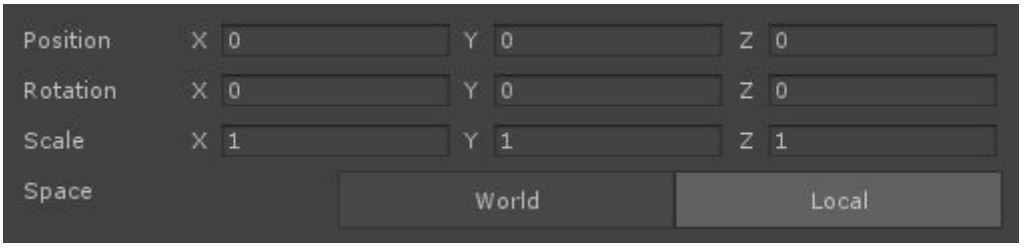
The **Shape** option will let you decide of the volumetric shape of the injection volume. The available shapes are **Global, Box, Layer, Sphere, Cylinder and Cone**. The **Fade** options are the normalized percentage of the different ways of fading the volumes on its borders. These settings vary with the selected shape. The **Fade Falloff** is the curve of the fading.

The **Noise Mask** option will allow you to generate a dynamic noise to be used as a mask by the injected data.

The **Texture2D Mask** option will allow you to set a Texture2D to be used as a mask by the injected data.

The **Texture3D Mask** option will allow you to set a Texture3D to be used as a mask by the injected data.

You will be able, for each mask, to set its own position, rotation and scale, in local or global space.



- **Inject Data** panels
  This panel contains the data you will be able to inject and their parameters. These data will be masked out by the **Volume Mask**.
  All of these injected data can be set as negative, meaning that you can actually remove this data. E.g. Set a volume inside a house with negative density injection will remove density and will result in a house with less density.

The **Density** injection will let you add or remove fog density inside the volume

The **Light** injection will let you add or remove a plain light inside the volume

The **Scattering** injection will let you modify the scattering factor inside the volume

The **Ambient Lighting** injection will let you add or remove ambient lighting (the one from the Lighting window) inside the volume

The **Light Probes Lighting** injection will let you add light probes lighting inside the volume

Illuminate/Fog Particles

A collection of shaders to be set on particles is provided with Aura 2.
These allow to illuminate/fog the particles with the volumetric lighting computed by the Aura system.

If you want to illuminate/fog your particles with Aura, you can choose between the provided shaders or make your own, following the provided ones as examples.

Amplify Shader Editor Nodes

Custom Amplify Shader Editor Nodes are provided to easily implement Aura in your shaders made with Amplify.

More nodes will be gradually added, allowing to apply Aura in different ways on different types of shaders.

Using Aura in your Shaders

The volumetric illumination computed by Aura is globally available and can be easily retrieved to lit objects.

Be aware that this illumination doesn't take into account any BRDF nor Lambertian term.

Aura can be implemented in your own custom shaders (transparent or not) in 3 very piece-of-cake steps.

1. Include "Aura.cginc" located in the "/Aura 2/System/Code/Shaders/" folder.
   #include "Assets/Aura 2/System/Code/Shaders/Aura.cginc"

2. In the Vertex Shader, compute the position inside the Aura frustum with
   float3 Aura2_GetFrustumSpaceCoordinates(float4 inVertex)

3. You can now apply the volumetric lighting with the following method
   void Aura2_ApplyLighting(inout float3 colorToApply, float3 screenSpacePosition, float lightingFactor)
   and/or apply the fog with the following overloads
   void Aura2_ApplyFog(inout float3 colorToApply, float3 screenSpacePosition)
   void Aura2_ApplyFog(inout float4 colorToApply, float3 screenSpacePosition)

Requirements

Aura 2 strictly requires full support of the following elements to work :

- RenderTextures (3D as well)
- Texture2DArrays
- ComputeShaders

Please verify that the support of these elements is not limited especially on lower platforms.

Aura 2 release was targeted for Unity 2017.2 :
- older version will not be supported
- newer version will be supported with updates if necessary

Aura 1 and Aura 2 are not compatible with each others.
Please delete all references to Aura 1 before importing Aura 2 in your project.

Acknowledgement

Aura is inspired/uses the following works :

- Bartlomiej Wronski (@BartWronsk)'s presentation : "Volumetric Fog : Unified compute shader based solution to atmospheric scattering"
- Ashima Arts's 4D Simplex Noise : https://github.com/ashima/webgl-noise

About the provided materials :

- Unity Lab's smoke spritesheet : "VFX Image Sequences & Flipbooks"

Special Thanks

For their time and help, I would like to cheerfully thank :

- Bartlomiej Wronski (@BartWronsk)
- All the people that helped me by testing Aura 1 and Aura 2, and kept me motivated with their constructive feedbacks and their kind words.

License

Aura 2 is a commercial project and is **not** in the public domain.
The intellectual and technical concepts contained in this package are proprietary to Raphaël Ernaelsten and are protected by copyright laws.
Dissemination of this information or reproduction of this material is strictly forbidden.

Aura 2 is under the EULA of Unity's Asset Store.

Changelog

2.0

Initial release

2.0.1

Additions

- *[Editor]* Added options to hide/show gizmos in the scene view when objects are selected/unselected

Changes

- *[Editor]* Enabled toolbox's notifications by default
- *[Editor]* Temporarily disabled Aura buttons in Light/Camera components
- *[Examples]* Fixed missing references in example scenes
- *[Core]* Prevented potentially ambiguous call of XRSettings
- *[Editor]* Fixed Directional Light gizmo

2.0.2

Additions

- *[Editor]* Added an option to enable/disable toolbox's animations

Changes

- *[Core]* Fixed blocking issue with multi-camera volumetrics rendering
- *[Editor]* Fixed issues with user-interface float fields not allowing dragging to smoothly modify values
- *[Examples]* Removed unnecessary lightmaps from the Directional Light example scene

2.0.3

Additions

- *[Core]* Added an option to **inject Ambient Lighting** with the Aura Volumes

Changes

- *[Editor]* Fixed blocking in-editor issue resulting in being unable to use transformation tools on game objects until reboot of the editor
- *[Editor]* Improved in-editor performances

- *[Core]* Fixed issue where post-processes might not render correctly if executed before Aura 2
- *[Doc]* Several additions and corrections to the documentation
- *[Editor]* Fixed typos in the toolbox
- *[Editor]* Fixed float fields misalignment for the output range values in the Level parameters of Aura Volumes

2.0.4

Additions

- *[Core]* Added color temperature mode support for lights
  ATTENTION : Unity currently gives no way to know if the color temperature mode is enabled on the light, therefore you will need to manually enable it in the Aura Light component

2.0.5

Additions

- *[Shaders]* Added new "Unlit Textured" standard shader that supports Aura Illumination and Aura Fog.
  This shader is made with Amplify Shader Editor so you can use it as an example.

Changes

- *[Core]* Reduced heap allocation and stalls due to memory garbage collection/flush
- *[Core]* Fixed occasional shader compilation error, not supporting structs methods declaration
- *[Editor]* Custom color circular picker now uses Unity's internal copy/paste methods and is now able to copy from/paste to Unity's regular color fields

2.0.6

Changes

- *[Core]* Fixed an issue with camera matrices in single-pass stereo mode

2.0.7

Additions

- *[Editor]* Added options to selectively display gizmos on Cameras, Lights and Volumes

Changes

- *[Core]* Fixed blocking issue of point lights' shadows not working in builds. Worked around Unity not properly setting shader keywords in player builds
- *[Shaders]* Fixed Texture2DArray shader model compatibility issues on provided particles shaders

2.1

Notes

- **<u>Quality and performances will be highly improved</u>**
- **Occlusion Culling is no longer experimental**

Additions

- *[Core]* Added support for **Sprite Renderers** (AuraSprite component)
- *[Core]* Added support for **Orthographic Cameras** (2D Cameras)
- *[Core]* Added option to **ignore scattering** globally or per light
- *[Core]* Added option to **override scattering** per light
- *[Core]* Added option to **tint lighting** globally or per volume
- *[Core]* Added option to **boost** the strength of the incoming **lighting** inside volumes
- *[Core]* Added option to **blur** the volumetric lighting
- *[Core]* Added option to debug occluded cells
- *[Editor]* Added small **add/toggle shortcut buttons** next to game objects in the hierarchy view
- *[Editor]* Added assembly definitions

Changes

- *[Core]* **Removed scattering bias on lights**. Scattering can now be overridden per light
- *[Core]* **Improved reprojection** where previous cell's state is invisible (reduced ghosting effect)
- *[Core]* Renamed "Aura 2/System/Code/Shaders/Shaders/System" folder into "Aura 2/System/Code/Shaders/Shaders/Core"
- *[Core]* Renamed "Aura 2/System" folder into "Aura 2/Core"
- *[Editor]* Fixed toolbox wobbling when reloading SceneView
- *[Editor]* Changed windows' IDs of Toolbox to maximize compatibility with other plugins
- *[Editor]* Fixed Toolbox related console's errors after removing Aura 2 from the project
- *[Editor]* Handled change of API from 2019+
- *[Platforms]* Improved Vulkan, Metal and OpenGL compatibility. Still no formal support
- *[Core]* Fixed occasional shader compilation issue
- *[Editor]* Fixed preview button position and size for 2019.1+ and 2019.3+
- *[Doc]* Updated manual and API documentation

2.1.1

Changes

- *[Core]* Hotfix for platforms supporting less than 512 compute threads

2.1.2

Changes

- *[Core]* Maximized compatibility for non DX platforms
- *[Core]* Fixed compilation error cause by the removal of the occlusion highest accuracy setting
-

2.1.3

Changes

- *[Core]* Maximized compatibility for non DX platforms
- *[Editor]* Updated assembly definition files to avoid errors about unknown target platforms

Note : If you experience any problem with updating Aura 2, please delete the Aura 2 folder, reboot Unity then update/import the newest version from the Asset Store

2.1.4

Changes

- *[Core]* Maximized compatibility for non DX platforms
- *[Core]* Fixed occasional error with texture untagged as UAV when Occlusion Culling is disabled

2.1.5

Changes

- *[Core]* Fixed shaders using Aura system's illumination being colorless Please reimport the "Aura 2 /Core/Code/Shaders" folder

2.1.6

Changes

- *[Core]* Fixed compatibility for Metal (MacOSX, iOS)
- *[Editor]* Prevented serialization of generated render textures on Aura Lights
- *[Editor]* Improved performances of the Aura Camera custom inspector

Additions

- *[Editor]* Added option to display/hide the buttons in the hierarchy