Documentation

Raphael Ernaelsten

@RaphErnaelsten

Table Of Contents

About Aura

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).

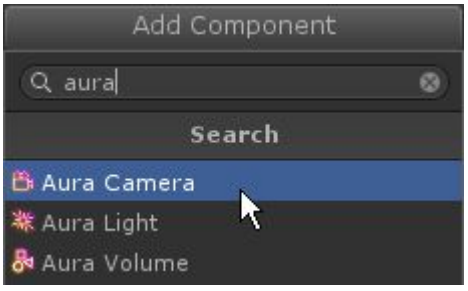
Aura is not an atmospheric scattering simulation nor a cloud simulator.

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 :

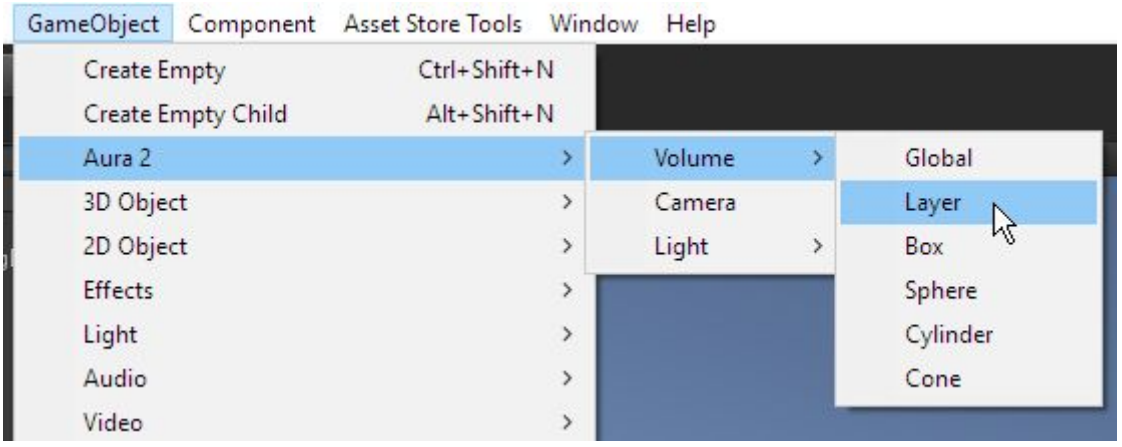
use the Add Component button at the bottom of the components pile of your GameObject



use the shortcut buttons integrated in the Toolbox

use the Add Aura button on Camera/Light GameObjects

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 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 two panels :

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

- **Directional / Spot / Point Parameters** panels
  These panels contain settings that are unique to the type of the light.

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**.

- **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**.

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/Shaders/" folder.

```
#include "../../../Aura/Shaders/Aura.cginc"
```

2. In the Vertex Shader, compute the position inside the Aura frustum with
   void Aura_ApplyLighting(inout float3 colorToApply, float3 screenSpacePosition, float lightingFactor)

```
o.frustumSpacePosition = Aura_GetFrustumSpaceCoordinates(v.vertex);
```

3. You can now apply the volumetric lighting with the following method
   void Aura_ApplyLighting(inout float3 colorToApply, float3 screenSpacePosition, float lightingFactor)

```
Aura_ApplyLighting(color, i.frustumSpacePosition, 1.0f);
```

   and/or apply the fog with the following overloads
   void Aura_ApplyFog(inout float3 colorToApply, float3 screenSpacePosition)
   void Aura_ApplyFog(inout float4 colorToApply, float3 screenSpacePosition)

```
Aura_ApplyFog(color, i.frustumSpacePosition);
```

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

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

Changes

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