The Experimental DXR project is a sandbox where you can test real-time ray tracing features in Unity. Be aware that this is a prototype and the final implementation of DXR will differ from this version. You can not use this project for any production work.

Project zip with all files (including LFS files) can be downloaded from the release section https://github.com/Unity-Technologies/Unity-Experimental-DXR/releases.

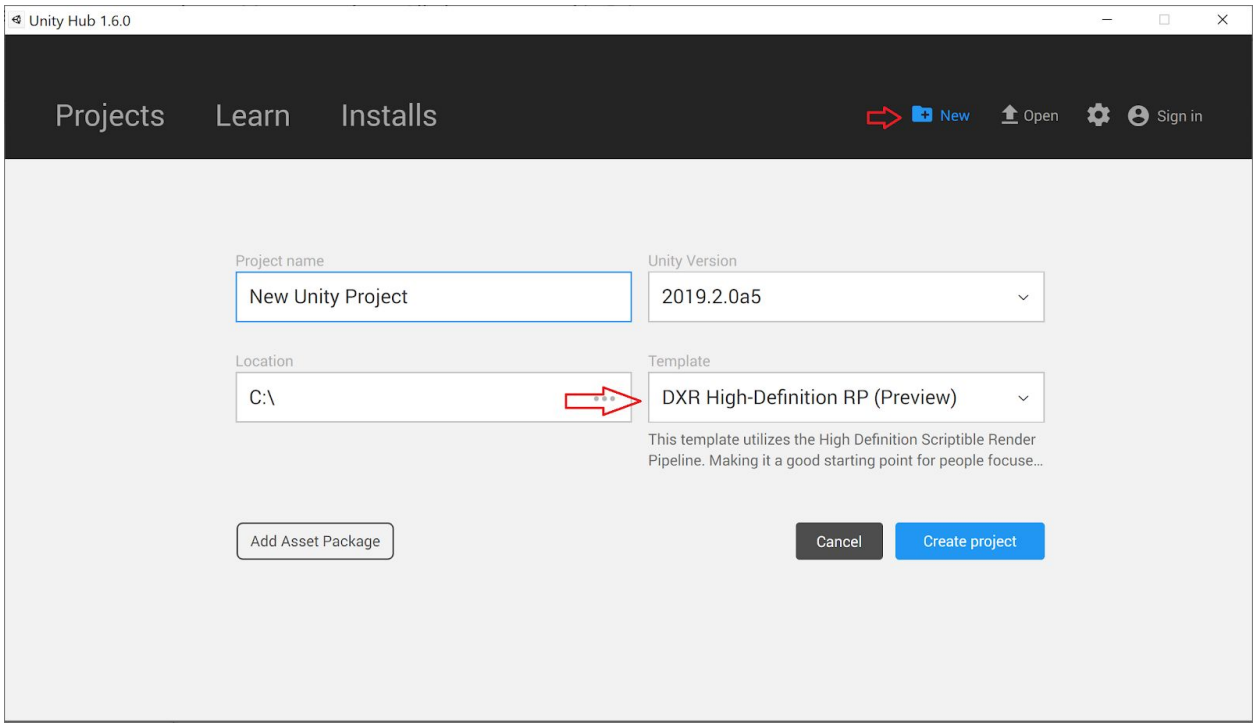
This project itself uses a custom version of Unity with binaries based on Unity version 2019.2a5. It includes version 5.8.0 of the High Definition Render Pipeline enhanced with DXR support. It is a Windows only version that uses DX12 API.

Requirements:

- NVIDIA RTX series card with the latest drivers [here](https://www.nvidia.com/Download/index.aspx?lang=com)
- Windows 10 RS5 (Build 1809) or later

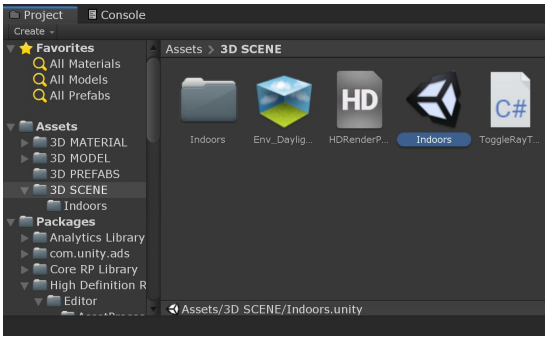
Install step:

- Download the project [here](https://github.com/Unity-Technologies/Unity-Experimental-DXR/releases) and unzip it.
- Launch Unity.exe
- Create a new project and select DXR High Definition RP (Preview)



This sets up the sandbox DXR project with the HDRP package.

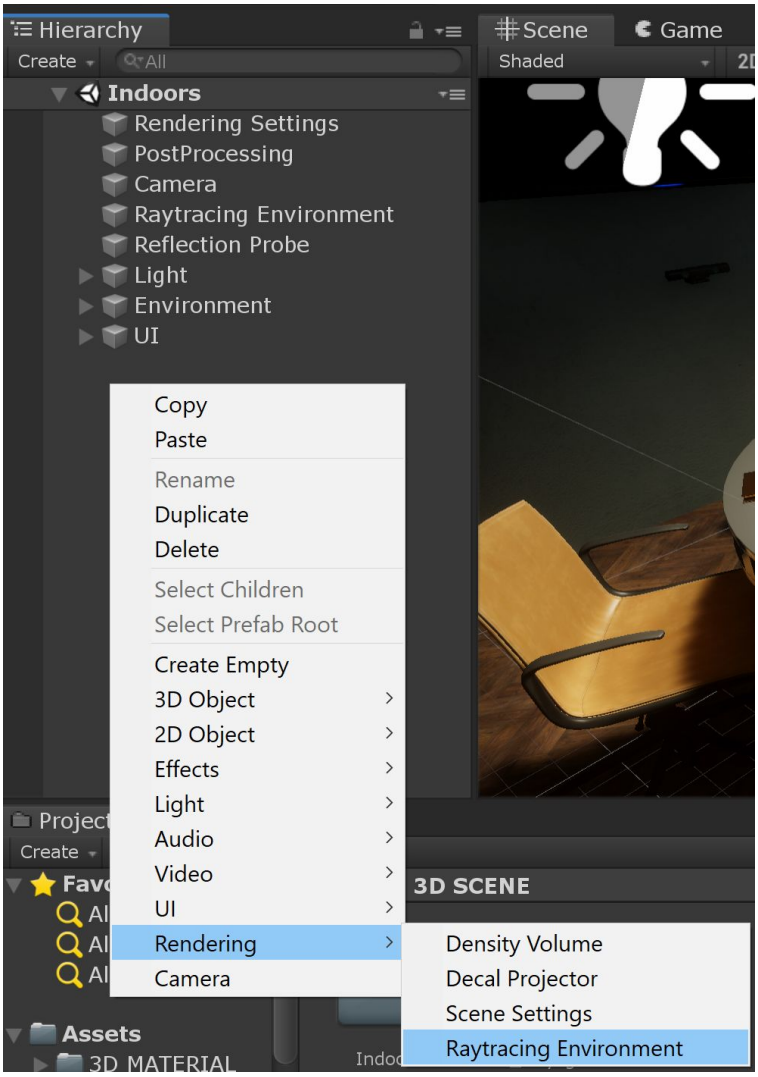
- In 3D SCENE folder, open the Indoors Scene



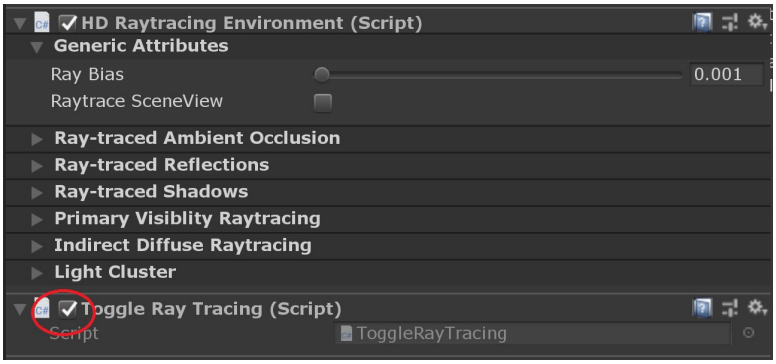
- Click the Play button to see the Scene. You can now enable/disable the ray tracing effects. (Note: by default, the ray tracing effects are only visible in game view).

Ray Tracing configuration:

- To get access to ray tracing features, you need to create a ray tracing environment. There is one already in the provided **Indoors** Scene.



- The ray tracing environment inspector allows you to configure the various effect of ray tracing and to disable/enable ray tracing globally



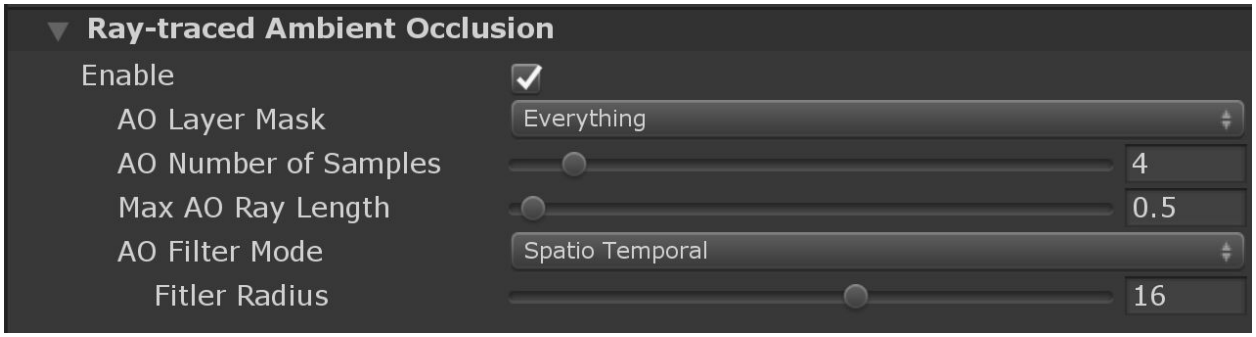
Ray Bias: Allows you to shift the start of a ray to avoid self intersection.
Raytrace SceneView: Allows you to enable the ray tracing effect in the Scene view, otherwise the ray tracing effects are only visible in the game view. It is often easier for artists to navigate in

a Scene without ray tracing, particularly when Scene you uses ray tracing with is quite resource intensive.

Other settings allow you to configure each effect. In each of these effects, there is a LayerMask which allows you to control which GameObjects the effect takes into account. If a GameObject is not in the same Layer as an effect, then the effect has no impact on it.

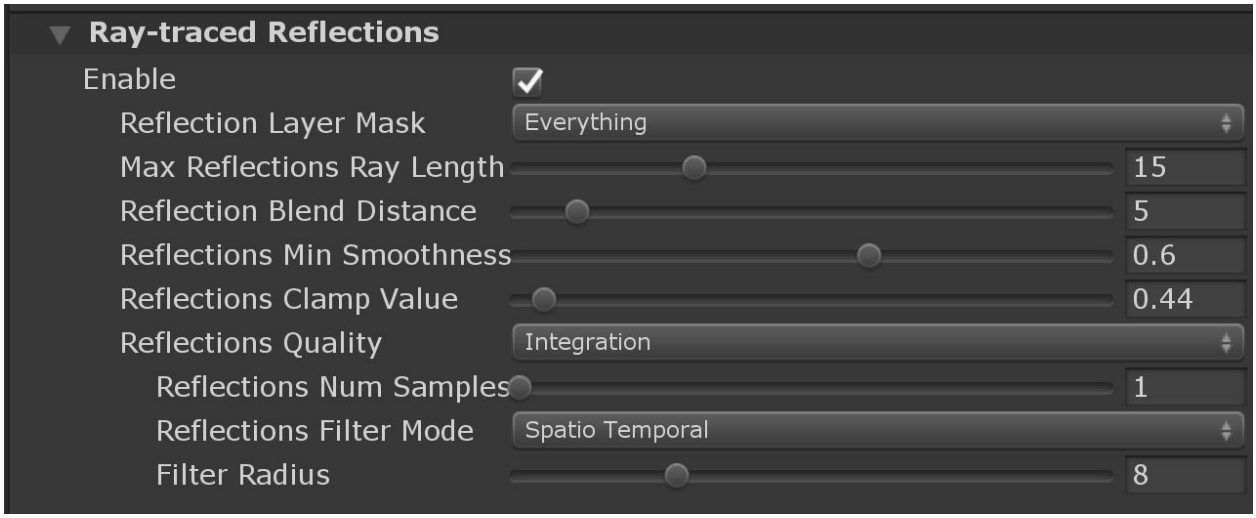
Ray Tracing Effects

Ray Traced Ambient Occlusion



AO Layer Mask: Enables control of which GameObjects this effect interacts with.
AO Number of Samples: A higher number of samples gives better quality, but also increases the resource intensity of the effect. You should use four samples for real-time performance.
Max AO Ray Length: Allows control of the maximum length that a ray can travel before being considered as a miss. A low value increases performance.
AO Filter Mode: Denoiser type: None, Spatio temporal or NVIDIA denoiser. The following settings depends on the chosen denoiser and allows you to control the quality of it at varying cost.

Ray Traced Reflection



This effect allows to add reflections to your Scene and will replace screen space reflection if enabled.

Reflections Layer Mask: Allows control of which GameObjects this effect interacts with.

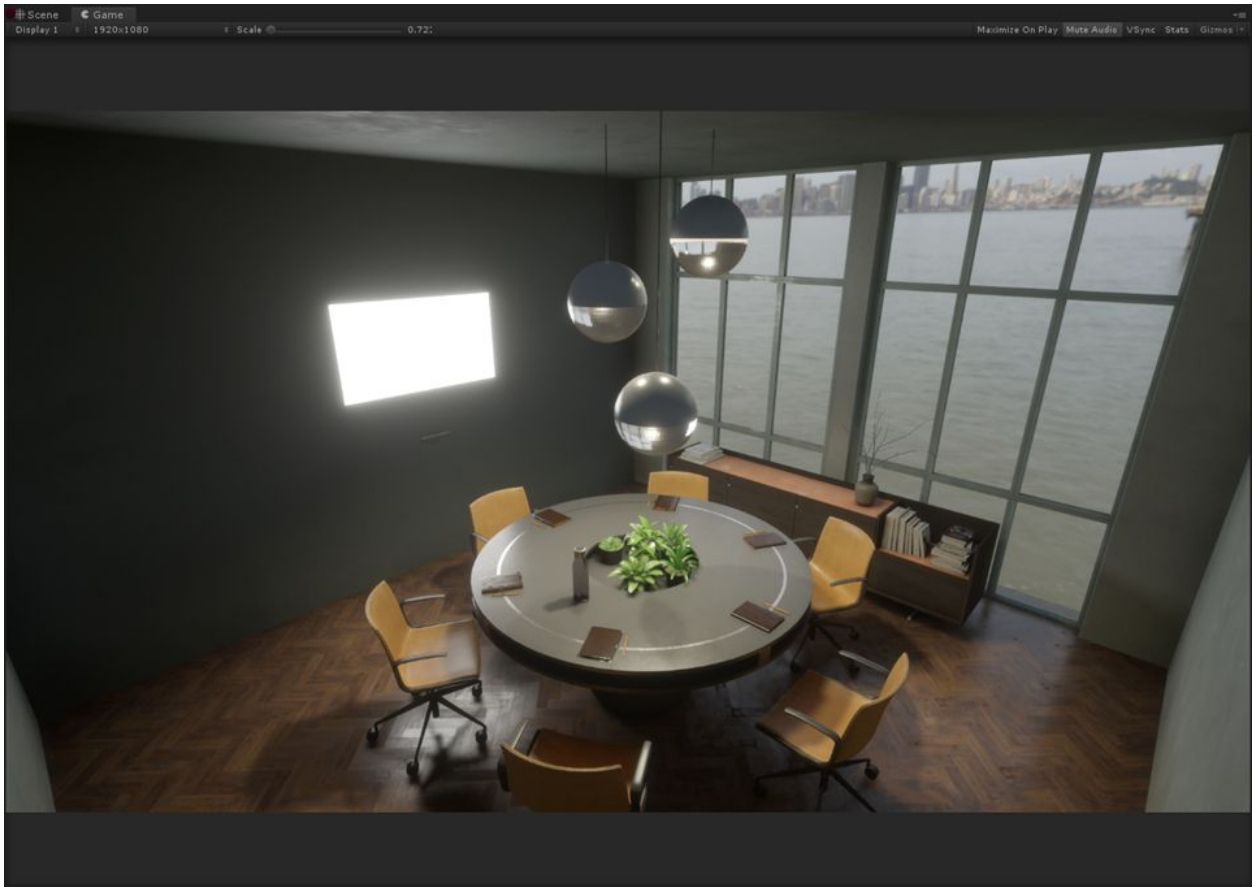
Max Reflections Ray Length: Allows control of the maximum length that a ray can travel before being considered as a miss. A low value increases performance.

Reflections Blend Distance: Unused.

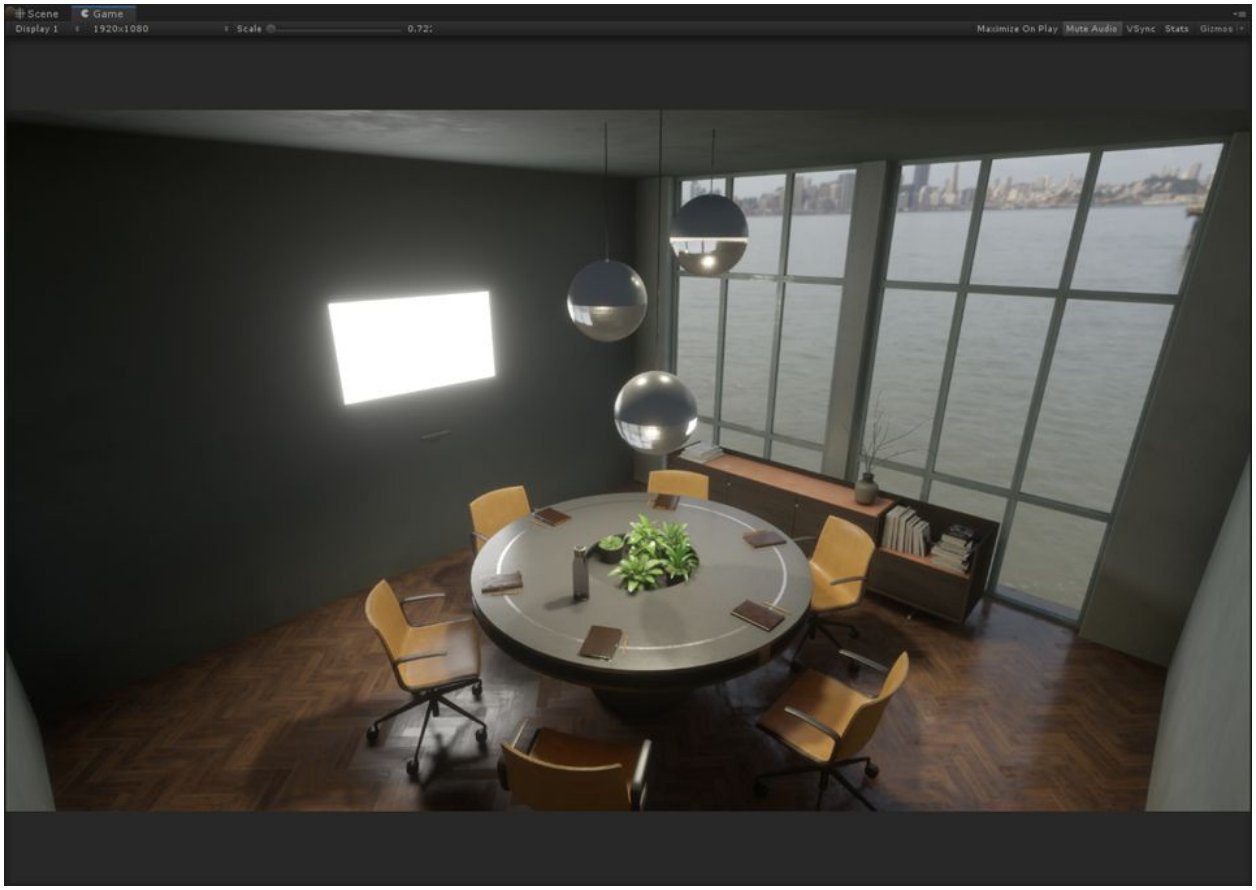
Reflections Min Smoothness: The minimum smoothness a Material must have for this component to process ray traced reflections. If a Material has a lower smoothness than this value, then it falls back to Reflection Probes or the sky.

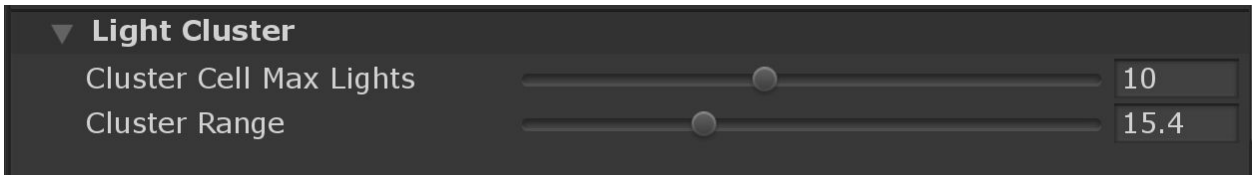
Reflections clamp values: Allows you to define, in pre-exposed space, a threshold to reduce [fireflies](https://en.wikipedia.org/wiki/Fireflies_(computer_graphics)). Try to keep this value above 1 to keep the bloom effect on intense values.
Reflections quality: QuaterRes (Reflection is rendered with one sample per pixel (so ¼ for full screen) with temporal accumulation) or **Integration** (Reflection is render full screen with number of sample ray per pixel). In both cases, Unity applies a denoiser at the end. This denoises has a controllable filter radius. A larger radius increases resource intensity.

Example of Integration (full resolution) result:



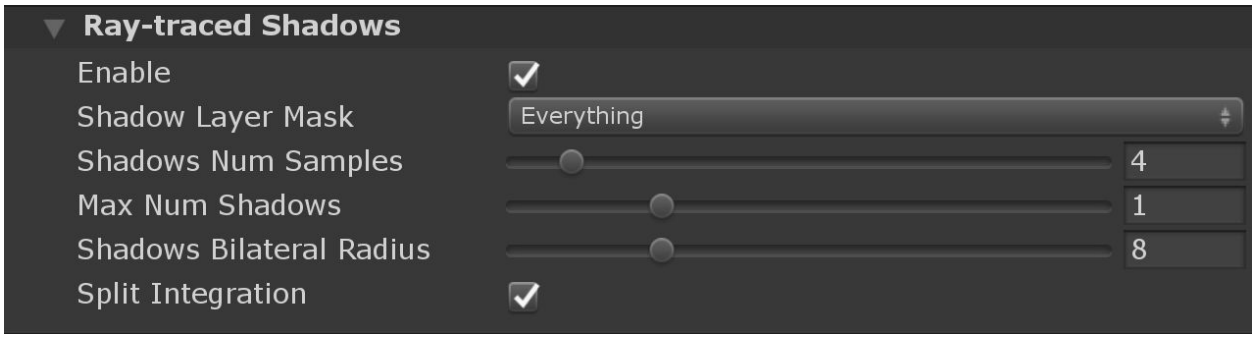
Example of quarter resolution result. Difference is barely noticeable for a good improvement in performance:





The reflections can reflect lights and GameObjects outside of the viewing frustum. To deal with this limitation, Unity generates a light cluster and forwards it to the ray traced shader. The light cluster is a voxel of 64x64x32 with a size defined by **Cluster Range** (Half Extent of Voxel around the Camera center) and a maximum number of light per cell of **Cluster Cell Max Lights**.

Ray Traced Shadow for Rectangular Area Light



These settings allow you to enable ray traced area shadows for the Textured rectangular area light.

Shadow Layer Mask: Allows control of which GameObjects this effect interacts with.

Shadow Num Samples: A higher number of samples gives better quality but also increases resource intensity. You should use 4 samples for real-time performance.

Max Num Shadows: Number of ray traced area shadows allowed into the Scene. Once Unity reaches the maximum, it falls back to having no shadows for the extra Lights.

Shadow Bilateral Radius: Control the size of the blur kernel the denoiser uses. Higher values help to reduce noise but increase the resource intensity of this effect.

Split Integration: When enabled, Unity processes the shadow samples one after the other instead of all at the same time. This improves the performance with large area Lights. You should measure the Impact of this option to make sure that it is worth it for your application because it doesn't change the quality.

Primary Visibility Raytracing

This effect allows you to render GameObjects with ray tracing. Unlike other effects that rely on secondary ray (i.e the first ray is solve with rasterization), this effect is aims to render smooth transparent GameObjects even if it works on opaque GameObjects. This feature only supports smooth GameObjects (no rough refraction).

Primary Visibility Layer Mask: Allows control of which GameObjects this effect interacts with.
Raytracing Maximal Depth: Defines the number of times a ray is allowed to trigger another ray when it intersects with aGameObject. This allows you to have recursive smooth reflection and refraction.

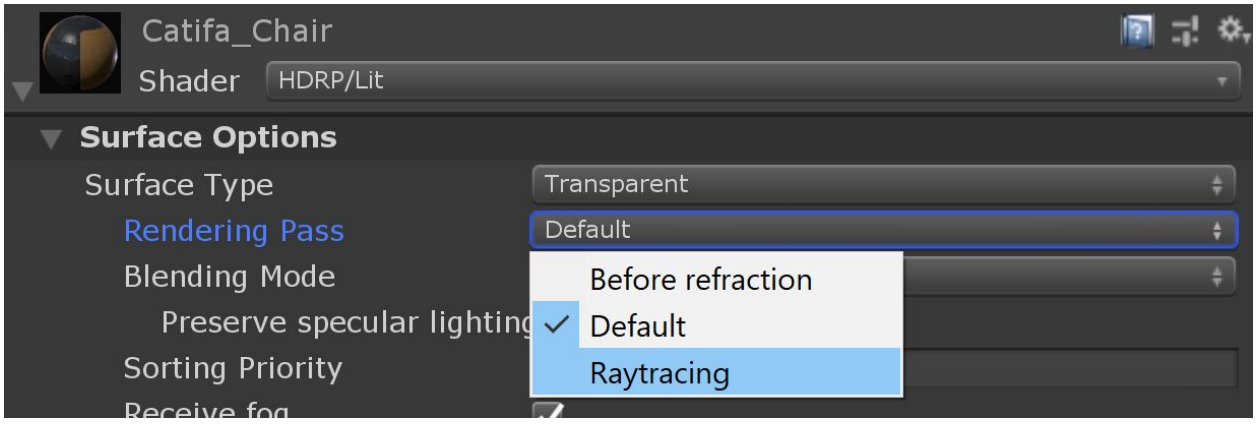
This example shows two glasses rendering with a **Maximal Depth** of 1:

This example shows two glasses rendering with a **Maximal Depth** of 5:

This example shows two glasses rendering with a **Maximal Depth** of 7:

Raytracing Ray Length: Allows control of the maximum length that a ray can travel before being considered as a miss. A low value increases performance.

To render a GameObject with the Primary-Visibility effect, edit the Material of the GameObject and select **Raytracing** from the **Rendering Pass** drop-down.

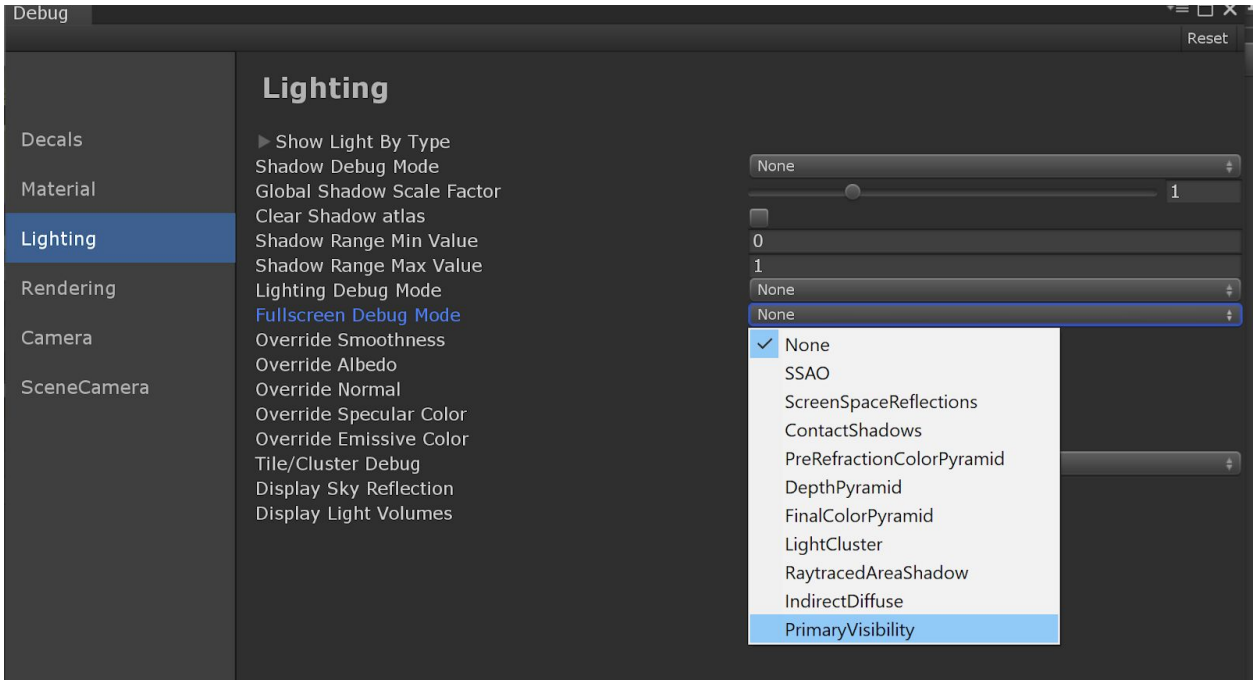


Indirect Diffuse Lighting

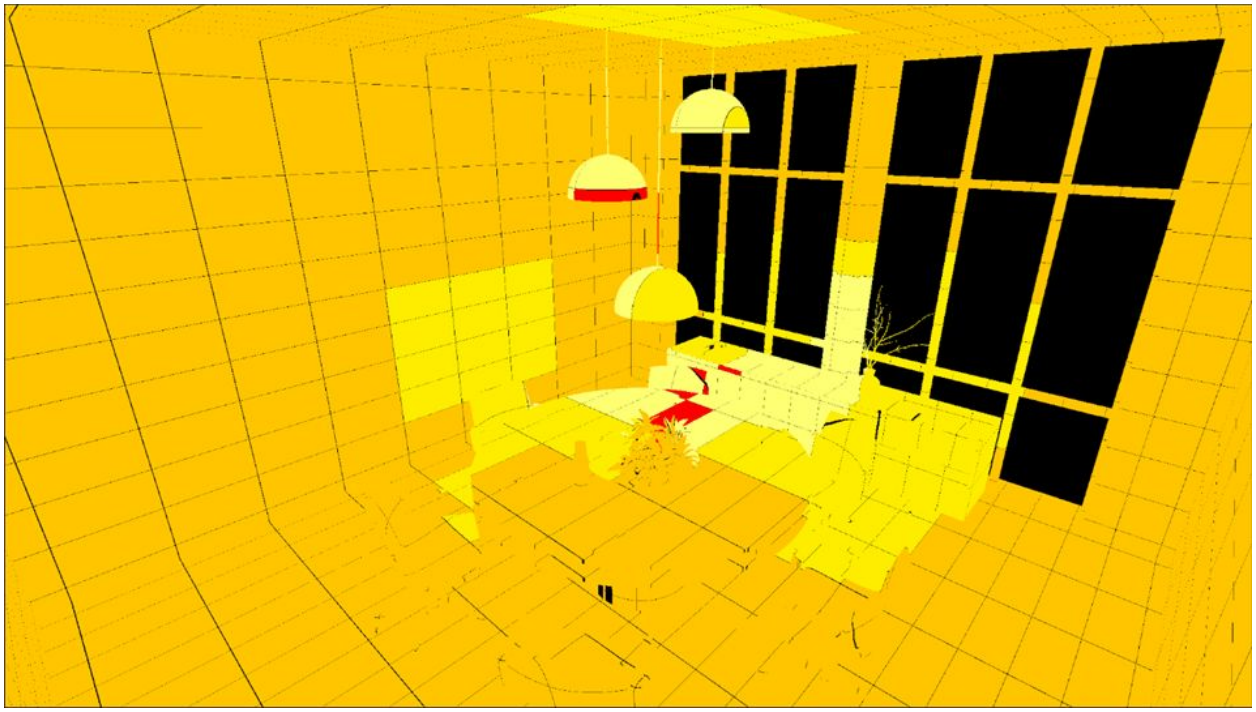
This effect allows you to add one diffuse bounce lighting to your Scene. This is a very early and brute force implementation and you can not use it for real time purpose. It is more for reference.
Indirect Diffuse Layer Mask: Allows control of which GameObjects this effect interacts with.
Indirect Diffuse Num Samples: A higher number of samples gives better quality but also increases cost. You should use at least 16 samples to get approximation of bounce lighting.
Indirect Diffuse Ray Length: Maximum distance that a ray travels before falling back to the sky lighting. Lower values increase performance.
Indirect Diffuse Ray clamp values: Allows you to define, in pre-exposed space, a threshold to reduce [fireflies](https://en.wikipedia.org/wiki/Fireflies_(computer_graphics)). Try to keep this value above 1 to keep the bloom effect on intense values.
Indirect Diffuse Filter mode: Use spatio temporal filter to get a noise free result.

Debugging

You can inspect the results of various effects with the debug menu. Windows -> Analysis -> Render Pipeline Debug. Select Lighting then choose one of the fullscreen debug modes:



- **SSAO** allows you to see the ray traced AO effect
- **ScreenSpaceReflections** allow you to see the ray traced reflection effect
- **Indirect Diffuse** allows you to see the indirect diffuse effect
- **PrimaryVisibility** allows you to see the primary visibility mask
- **LightCluster** allows you to see the light present in the light cluster



Light cluster: The color represents the number of lights <accumulation of yellow> hitting the area, Red displays the maximum budget for the lights

Material authoring

This experimental version only supports Lit and Unlit Material for ray tracing. Also, it only supports the HDRP/Lit ShaderGraph version.
This version does not support mipmap LODs when using a Material for ray tracing.

This version also only supports the **Deferred Only Lit Shader Mode** in the HDRP Asset.