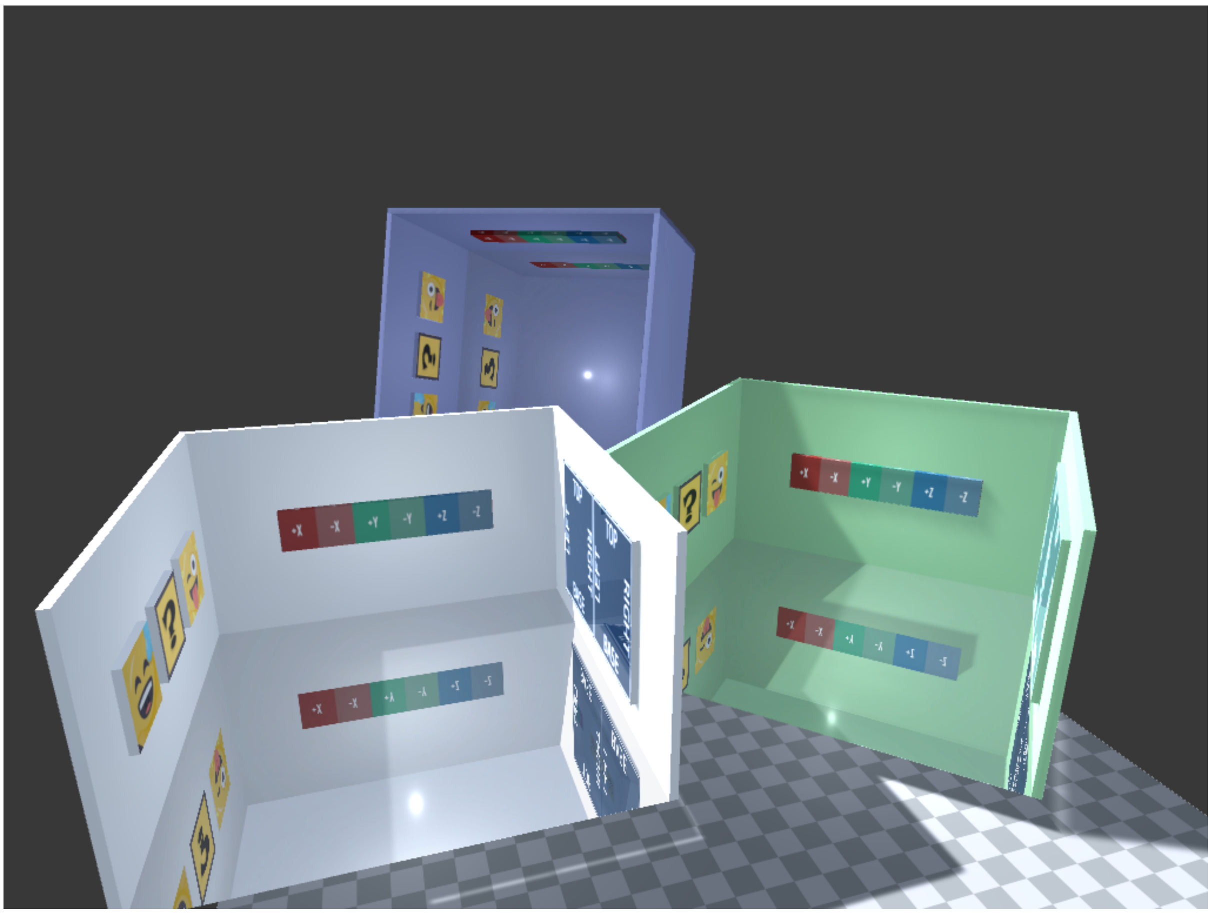
OOBB Reflection Probe Support Guide

Version: 0.0.6



Getting Started

Please read the following documentation in order to understand the limitations, support and usage of this package..

Current Support

Supported

- Forward and Legacy Deferred.
- Renderer.Probes.ReflectionProbes.Simple mode. May require probes to be anchored to the renderers.
- Single oriented bounding box reflection probe in a scene (simple use case).
- Multiple oriented bounding box reflection probes (complicated use case).

Unsupported

- Deferred.
- Renderer.Probes.ReflectionProbes.BlendProbes mode.

Settings

In the resource folder you'll find a 'OOBB Projection Settings' scriptableObject. You use this to change various options for the OOBB Projection.

Currently the only setting is to log debug statements to the console. Enabling this will generally output a large number of debug statements. This will create garbage that needs to be collected periodically and affect performance. Generally developers should not need to enable this setting.

Single OOBB Reflection Probe Per Scene

If you only need a single OOBB Reflection Probe in the scene then the simplest method is to use the ‘Single OOBB Per Scene’ method. This script assigns the required OOBB matrix via the use of Shader globals that make the property available to any material/shader in the project that requires it. This method requires minimal input from the developer and is easy to use.

For the single Reflection Probe that is to be used as a source for OOBB projection, assign the ***OOBB Probe Scene Global Static*** (OOBBGlobalReflectionProbe.cs) to its GameObject. This can be accomplished via the Component Menu under ‘Rendering’ or the inspector ‘AddComponent’.

Any GameObject Renderer that is to use OOBB projection must use the supplied custom **Standard (OOBB Projection)** shader for its Material.

Advantages

- Does not require per renderer components added to gameObjects.
- Does not require additional code or actions from the developer.

Disadvantages

- Does not allow more than a single OOBB in the scene or through additive scenes.

Multiple OOBB Reflection Probes

Supporting multiple OOBB Probes in a scene is a little more involved. It requires a custom component be applied to each Reflection Probe you want to perform OOBB projection and a renderer component added to every gameobject that is affected by an OOBB Reflection Probe.

For each Reflection Probe that is to be used as a source for OOBB projection, assign the ***OOBB Probe Controller*** component (OBBReflectionProbeController.cs) to its GameObject. This can be accomplished via the Component Menu under 'Rendering' or the inspector 'AddComponent'.

For each GameObject Renderer that is affected by a Reflection Probe and is to use OOBB projection, assign the ***OOBB Probe Renderer*** **OnEnable** component (OBBRenderer.cs) to the GameObject. This can be accomplished via the Component Menu under 'Rendering' or the inspector 'AddComponent'.

Any GameObject Renderer that is to use OOBB projection must use the supplied custom **Standard (OOBB Projection)** shader for its Material.

Advantages

- Supports multiple 'Simple' blend mode Reflection Probes in the scene.

Disadvantages

- More awkward to develop with and requires more input from the developer.
- Requires additional components per renderer.

Rotating an OOBB Probe

If you rotate a reflection probe, either via its own gameObject transform or a parent transform, you'll have to manually force all renderers to update their projection matrices. This can be accomplished by clicking on the 'Update All Renderers' button found in the Reflection Probe OOBB Controller component on the reflection Probe.

IMPORTANT

It's important to follow this ordering

- Rotate or move the reflection Probe gameObject or a parent.
- Selecting the Reflection Probe and click 'Update All Renderers'.
- Bake the reflection Probe.

This is important as if you use multiple bounces, failing to update the renderers first will result in bounced reflections being rendered incorrectly. You may not even notice this at first as the main reflection will likely be correct, but reflections of objects within that reflection may be incorrect.

If in doubt then you can perform the above sequence twice to be 100% sure the reflection is baked correctly.

Dynamic OOBB Reflection Probes

If you want to animate the reflection Probe in real-time you'll need to replace the ***OOBB Probe Renderer*** **OnEnable** component with the ***OOBB Probe Renderer*** **WillRender** component. This will ensure the renderers probe matrix gets updated every frame.

Hints & Tips

Bounced Multiple Reflections

Multiple reflections (e.g two mirrors facing each other) can be rendered through setting Lighting.EnvironmentReflections.Bounces to the number of reflections you require. For each bounce level the probe will be re-rendered when you bake it, meaning the first render will capture a single reflection in a mirror, whilst a second renderer will capture the scene and any previous reflection bake, etc.

Demo Scenes

You can install the demo scenes from the Unity Package Manager. Open up the package manager and select the OBB Reflection Probe package. You should then see a samples section along with an 'Import Into Project' button. Click the button and a samples folder will be imported into your assets.

The demo scenes are mostly built around testing and debugging but can be a useful learning tool.

Most demo scenes have a 'Collection' gameObject within the scene. This contains a component that allows you to switch between various scene set ups. Again this is mostly for debugging and checking that the OOBB version of an axis aligned or rotated scene looks the same as the default Unity Reflection Probe.

Issues

Newer version of Unity

The package is designed for 2019.4.28 as it includes Unity's built in shader code files from that version. While the package may work in newer versions of Unity it is more likely that the shader code files included will need to be updated and merged with any changes Unity has made. To do this on Windows I use WinMerge to compare files and see what changes have been made. In the case of the custom shader files I've tagged all OOBB code changes so they should be easy to find.

Don't forget you might also need to update the custom 'Standard (OOBB Projection) shader as well as the custom 'Internal-DeferredReflections' shader.

Performance

For multiple OOBB support every renderer requires an additional component which is far from efficient. Whilst the code currently avoids having to use Update method and the potential performance impact that would have for scenes with large numbers of gameobjects it is still not a satisfactory solution. This is even more true when we consider that in the vast majority of cases these renderers are likely going to be tagged as static and thus Unity will combine the meshes, but we'll still be executing at least OnEnable for all of them.

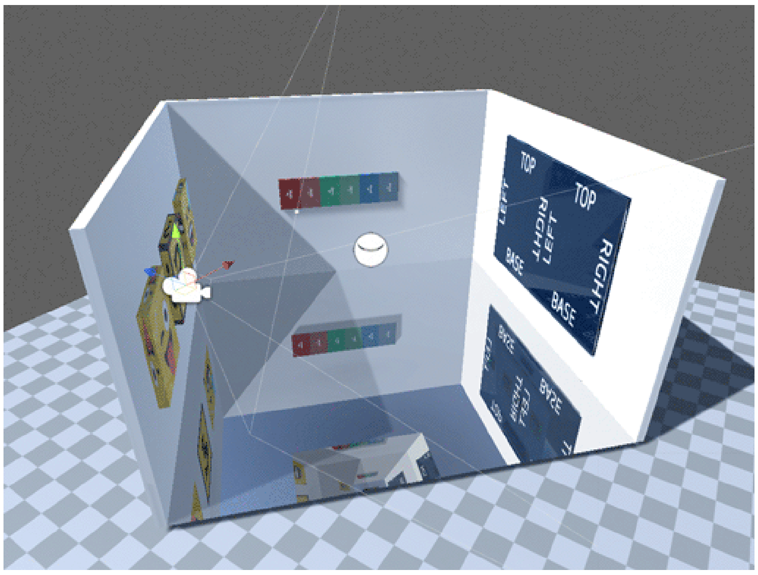
The obvious solution is a manger level component that can track which renderers are affected by which reflection probe. However this is not as straightforward especially when considering dynamic gameObjects and worse should we add dynamic reflection cubes into the mix.

The main problem is adding such a manager may lead to inconsistencies and may be a point of failure, particularly when developers have full control over their project. It's certainly something that should be implemented by developers who understand the needs of their project, but as a generalized solution its more of a challenge.

Deferred Rendering

While deferred rendering of OOBB Reflection Probes is currently not supported it has been explored. For single probes the OOBB reflection is rendered, but it appears to be clipped by the Unity Reflection Probe axis aligned bounding box.

To play around with deferred rendering you need to replace the default Unity Internal-DeferredReflections shader with the custom version. This is done via Project Settings - Graphics - Deferred Reflections.



FAQ

Models fail to be baked into reflection.
By default a ‘Baked’ Reflection Probe will ***only*** bake static geometry/models as such the majority of your scene and models should be set to static.

Non-static models can be baked but you have to switch the Reflection Probe Type to ‘Custom’ and enable ‘Dynamic objects’. Alternatively you can switch the Reflection Probe Type to ‘Realtime’ but this is only designed for situations where you need to rebake the reflection frequently at runtime.

My reflections look weird.

- Try forcing the recalculation and application of the OBB Reflection Probe Matrix.
- Try reloading the scene.Sometimes the active scene can become broken, especially if it has encountered a script error in editor mode. Since OBB Reflection Probe Support uses ‘Execute In Editor’ any script error can cause those scripts to stop working.

My reflections look slightly different to Unity’s AABB Reflection Probes?
Overall there should be no perceivable difference, but due to the fact you are rendering a cubemap at a rotation there will be some slight differences for example on straight lines as they may no longer be pixel aligned. It's only noticeable if you look hard enough and is often disguised due to bilinear filtering.

Do I need to rebake existing reflection probes?
No, you should not need to ***unless*** you are using bounced multiple reflections. For bounced reflections each bounce needs to ensure its using the OBB projection, so they will need to be rebaked.

Use of UNITY_MATRIX_MVP is detected.
You’ll see this error whenever the shaders for OOBB projection are rebuilt. Its safe to ignore as its coming from Unity’s own built-in shaders, the source of which are included in this package and referenced only by our custom **Standard (OOBB Projection)** shader.