Irreverent Software™

PLANAR REFLECTIONS™ 4

Index

PIDI – Planar Reflections 4
Introduction

PIDI Planar Reflections 4 is an advanced tool to add real-time reflections to your 3D scenes easily and with a low performance impact. The tool comes with dozens of custom shaders that allow you to create all kinds of reflective surfaces from simple mirrors and basic opaque surfaces to complex PBR materials, water and glass floors and walls.

Furthermore, the asset comes with specific shaders and workflows for mobile devices and the new Universal RP and HDRP pipelines for Unity 2019.4 and Unity 2020.3.

In this documentation you will find a general description of all the different features of this asset, a small setup guide and some performance tips to help you add reflections to your scenes in no time.

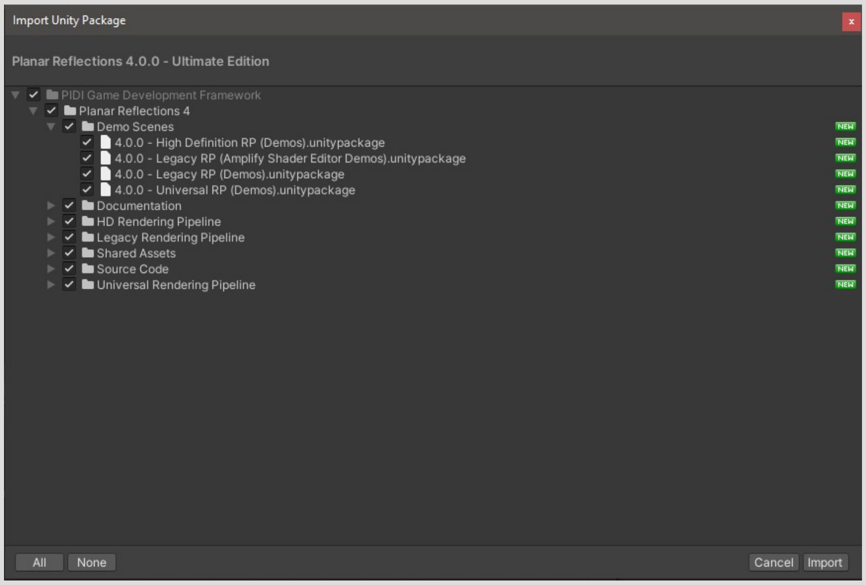
If you have any questions, suggestions or need support, contact us at support@irreverent-software.com

You can also find the online version of this documentation **here**

Getting Started

Installation

To install the asset, open the Package Manager and from the drop down menu select My Assets. Then, search for PIDI Planar Reflections 4. Download the asset and prepare to import it into your project. You will see the a screen showing all the contents of the asset (in the following picture we show the contents of the Ultimate Edition) :



Import Unity Package

Planar Reflections 4.0.0 - Ultimate Edition

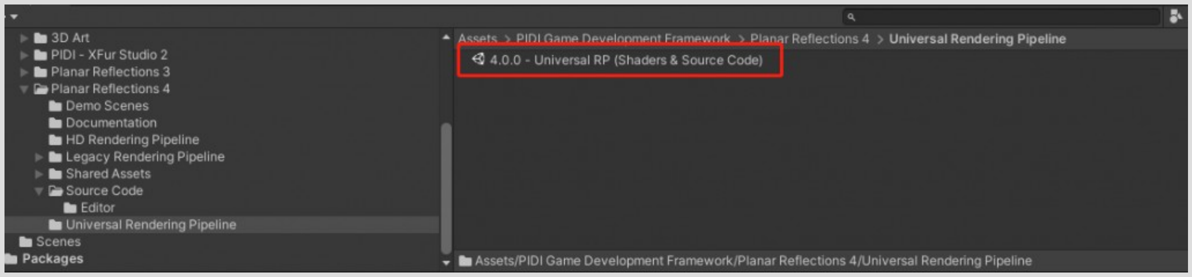
- PIDI Game Development Framework
  - Planar Reflections 4
    - Demo Scenes NEW
      - 4.0.0 - High Definition RP (Demos).unitypackage NEW
      - 4.0.0 - Legacy RP (Amplify Shader Editor Demos).unitypackage NEW
      - 4.0.0 - Legacy RP (Demos).unitypackage NEW
      - 4.0.0 - Universal RP (Demos).unitypackage NEW
    - Documentation NEW
    - HD Rendering Pipeline NEW
    - Legacy Rendering Pipeline NEW
    - Shared Assets NEW
    - Source Code NEW
    - Universal Rendering Pipeline NEW

All None Cancel Import

If you are using either the HDRP or Universal RP editions of the asset, you can simply import all the contents of the package into your project as there is no content from other pipelines available. If you are using the Ultimate Edition please read the following section.

Ultimate Edition and multiple rendering pipelines

If you are using the Ultimate Edition of the asset, remember to deselect all the folders designed for other pipelines to reduce the amount of content installed into your project and avoid any potential errors. Once you unpack the folder for the rendering pipeline you are using you can simply double click on the most recent unitypackage inside and this will automatically unpack and install all the corresponding files.



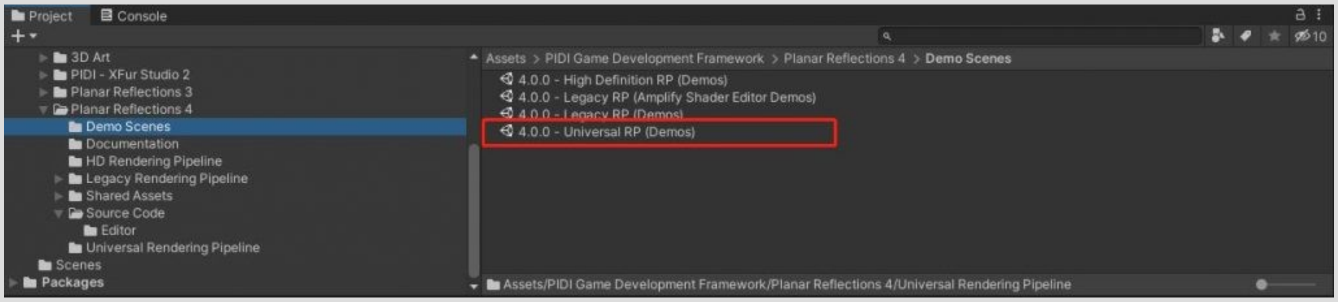
- 3D Art
- PIDI - XFur Studio 2
- Planar Reflections 3
- Planar Reflections 4
  - Demo Scenes
  - Documentation
  - HD Rendering Pipeline
  - Legacy Rendering Pipeline
  - Shared Assets
  - Source Code
    - Editor
  - Universal Rendering Pipeline
- Scenes
- Packages

Assets > PIDI Game Development Framework > Planar Reflections 4 > Universal Rendering Pipeline

4.0.0 - Universal RP (Shaders & Source Code)

Assets/PIDI Game Development Framework/Planar Reflections 4/Universal Rendering Pipeline

Always import the shaders and source code package first, followed by any additional resources you need, and finally install the demo packages. This is necessary to avoid any errors. Also remember that you can only have the contents for ONE pipeline installed at a time since the source code of the asset is different for each one of them.


Project
Console
3D Art
PIDI - XFur Studio 2
Planar Reflections 3
Planar Reflections 4
Demo Scenes
Documentation
HD Rendering Pipeline
Legacy Rendering Pipeline
Shared Assets
Source Code
Editor
Universal Rendering Pipeline
Scenes
Packages
Assets > PIDI Game Development Framework > Planar Reflections 4 > Demo Scenes
4.0.0 - High Definition RP (Demos)
4.0.0 - Legacy RP (Amplify Shader Editor Demos)
4.0.0 - Legacy RP (Demos)
4.0.0 - Universal RP (Demos)
Assets/PIDI Game Development Framework/Planar Reflections 4/Universal Rendering Pipeline

Every time you update the asset to a newer version, simply import the most recent unitypackage within the folder that matches the rendering pipeline you are using. We recommend you to back up older versions of the asset in case that you want to roll back any upgrade / update.

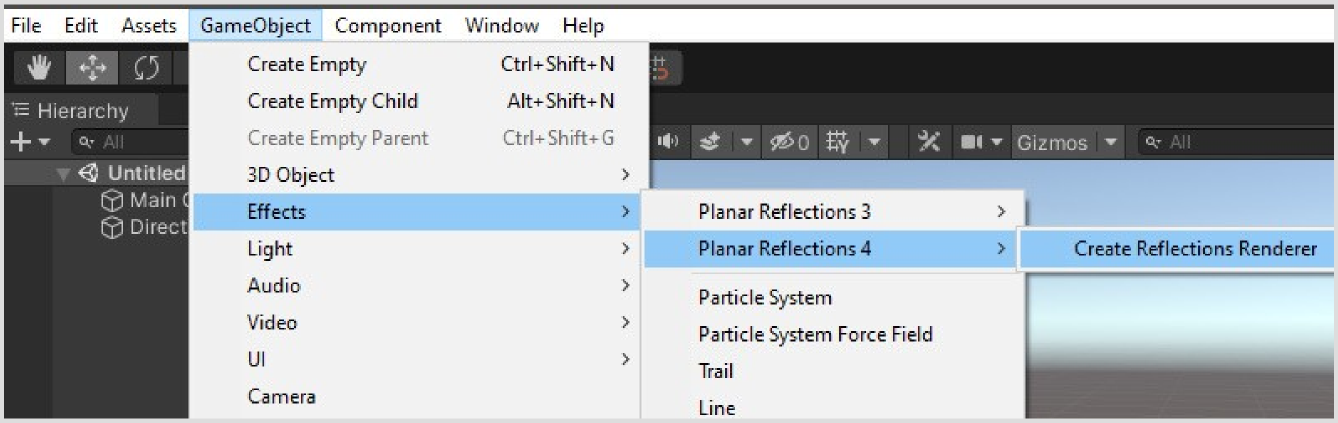
Please remember to make a backup of your project before upgrading or installing any tool or asset. While we do our best to ensure that our software is free of errors and easy to use, we are not responsible for any loss of data, corrupted files or projects produced during the installation or use of this software.

Once the asset has been successfully installed into your project, adding reflections to an existing scene is a simple process that can be done in just a few minutes. For more advanced uses and in-depth information about each feature please continue reading this documentation.

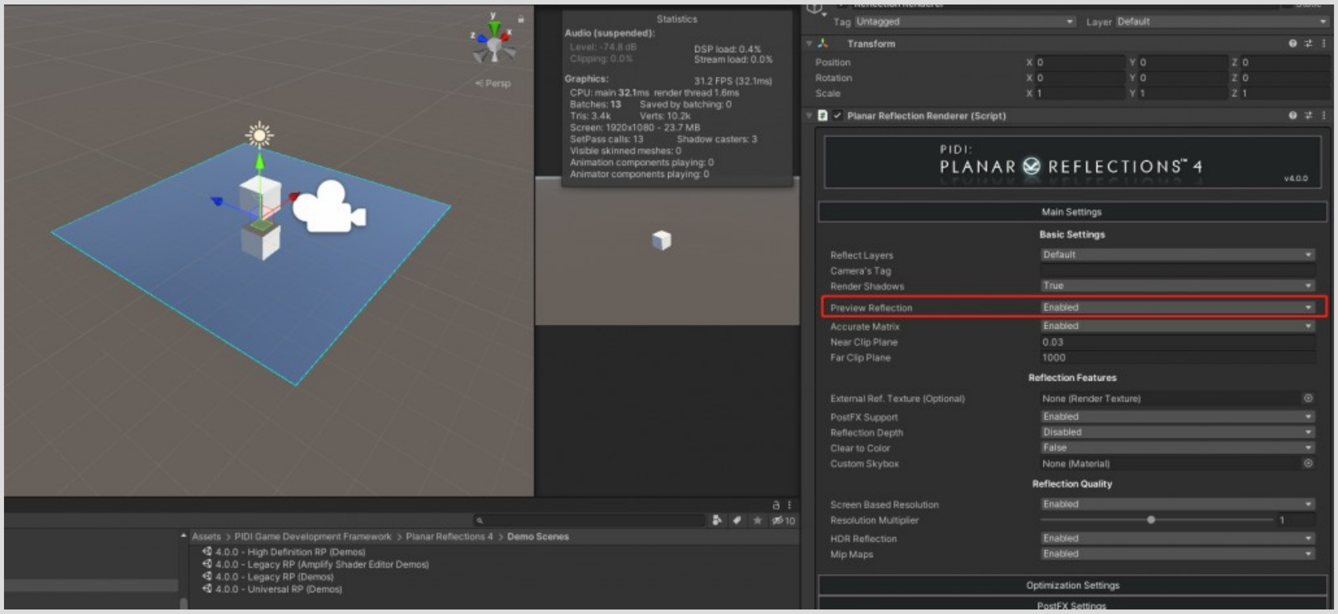
Adding Reflections to a Scene

Adding reflections to a scene with PIDI Planar Reflections 4.x is a very simple process that takes only a few minutes. To simplify the workflow of adding reflections in real-time and ensuring that they are as re-usable as possible within a level the process has been split into two main components, the **Reflection Renderer** which generates a reflection and its depth pass and a **Reflection Caster** which assigns it to a material and actually displays it in the scene.

To add a Reflection Renderer to the scene go to the GameObject tab on the top of the Unity Editor, then to Effects, Planar Reflection 4, Create Reflections Renderer.



This will create an empty reflection renderer in the middle of the scene. By default, it will not display the preview reflection renderer but you can enable it easily on the **Main Settings** tab of the Reflection Renderer component.



```
Statistics
Audio (suspended):
  Level: -74.8 dB                DSP load: 0.4%
  Clipping: 0.0%                 Stream load: 0.0%
Graphics:                        31.2 FPS (32.1ms)
  CPU: main 32.1ms  render thread 1.6ms
  Batches: 13       Saved by batching: 0
  Tris: 3.4k        Verts: 10.2k
  Screen: 1920x1080 - 23.7 MB
  SetPass calls: 13            Shadow casters: 3
  Visible skinned meshes: 0
  Animation components playing: 0
  Animator components playing: 0
```

Transform

Position	X 0	Y 0	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Planar Reflection Renderer (Script)

PIDI: PLANAR REFLECTIONS™ 4 v4.0.0

Main Settings

Basic Settings

Reflect Layers	Default
Camera's Tag	
Render Shadows	True
Preview Reflection	Enabled
Accurate Matrix	Enabled
Near Clip Plane	0.03
Far Clip Plane	1000

Reflection Features

External Ref. Texture (Optional)	None (Render Texture)
PostFX Support	Enabled
Reflection Depth	Disabled
Clear to Color	False
Custom Skybox	None (Material)

Reflection Quality

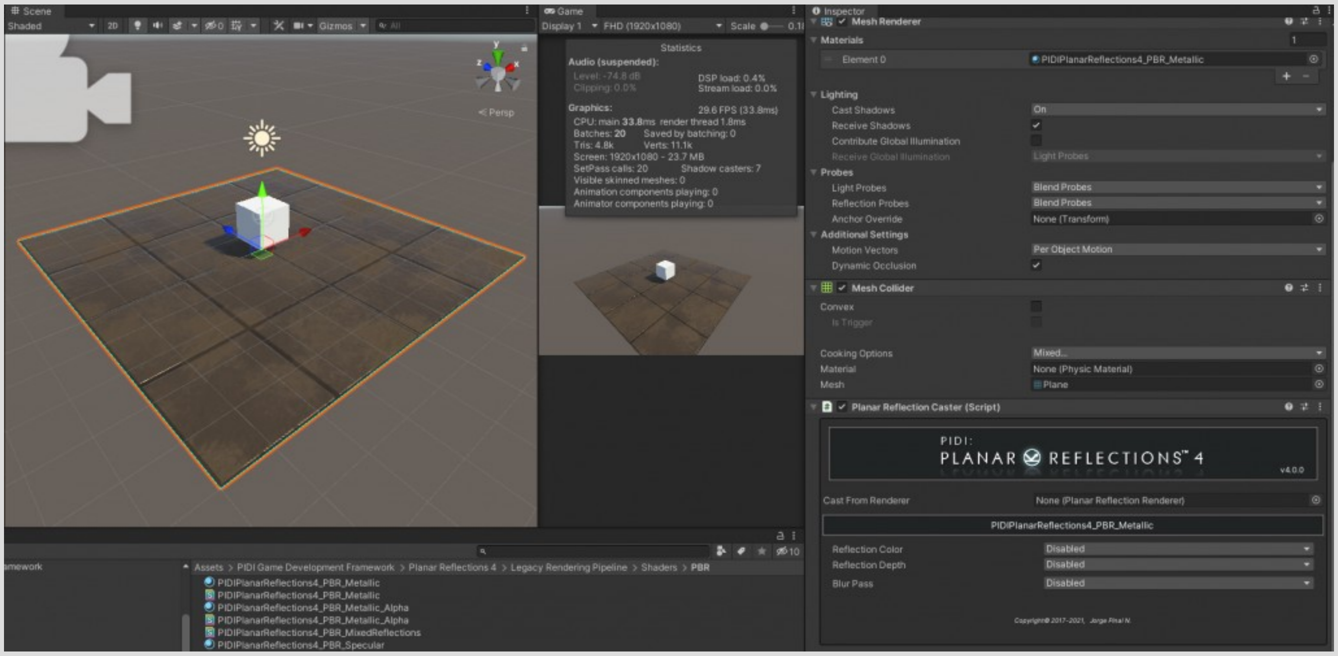
Screen Based Resolution	Enabled
Resolution Multiplier	1
HDR Reflection	Enabled
Mip Maps	Enabled

Optimization Settings

PostFX Settings

Persp
Assets > PIDI Game Development Framework > Planar Reflections 4 > Demo Scenes
4.0.0 - High Definition RP (Demos)
4.0.0 - Legacy RP (Amplify Shader Editor Demos)
4.0.0 - Legacy RP (Demos)
4.0.0 - Universal RP (Demos)

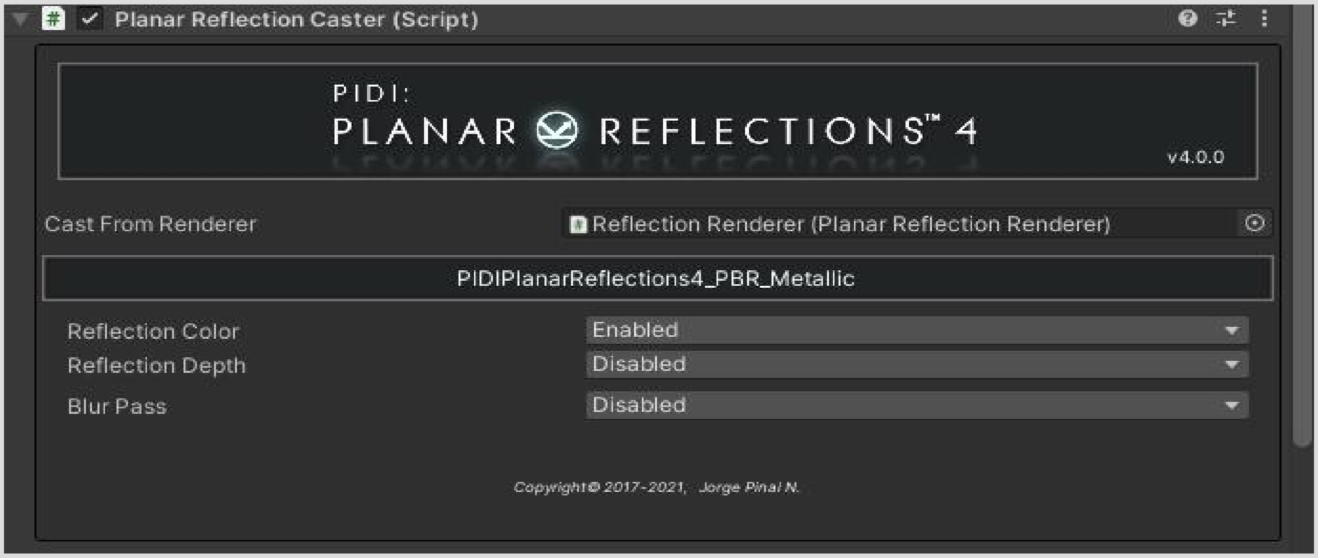
The reflection will not be visible at all on the Game View as it is not being displayed on any mesh at the moment. Let's add a plane to the scene and use one of the demo materials included with the asset on it. In our case, the **PIDIPlanarReflections4_PBR_Metallic** material. We will also add a Planar Reflection Caster component to it.


Statistics
Audio (suspended):
Level: -74.8 dB DSP load: 0.4%
Clipping: 0.0% Stream load: 0.0%
Graphics: 29.6 FPS (33.8ms)
CPU: main 33.8ms render thread 1.8ms
Batches: 20 Saved by batching: 0
Tris: 4.8k Verts: 11.1k
Screen: 1920x1080 - 23.7 MB
SetPass calls: 20 Shadow casters: 7
Visible skinned meshes: 0
Animation components playing: 0
Animator components playing: 0

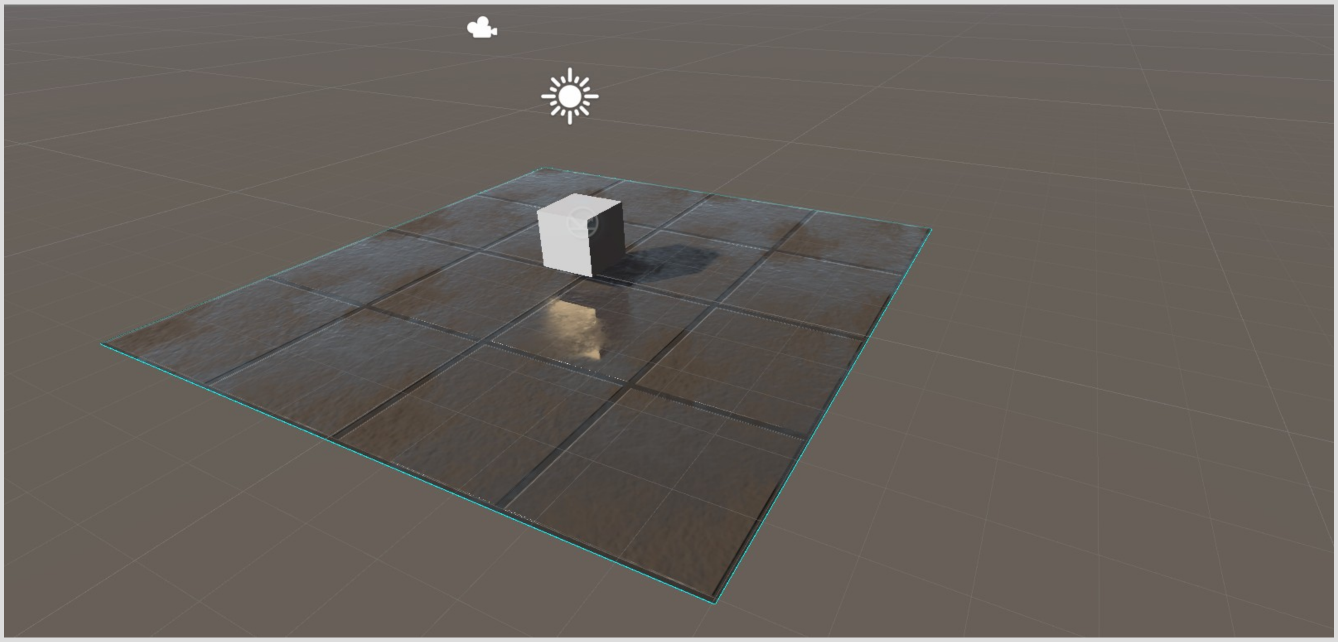
Mesh Renderer
Materials 1
Element 0 PIDIPlanarReflections4_PBR_Metallic
Lighting
Cast Shadows On
Receive Shadows ✓
Contribute Global Illumination
Receive Global Illumination Light Probes
Probes
Light Probes Blend Probes
Reflection Probes Blend Probes
Anchor Override None (Transform)
Additional Settings
Motion Vectors Per Object Motion
Dynamic Occlusion ✓
Mesh Collider
Convex
Is Trigger
Cooking Options Mixed...
Material None (Physic Material)
Mesh Plane
Planar Reflection Caster (Script)
PIDI: PLANAR REFLECTIONS™ 4 v4.0.0
Cast From Renderer None (Planar Reflection Renderer)
PIDIPlanarReflections4_PBR_Metallic
Reflection Color Disabled
Reflection Depth Disabled
Blur Pass Disabled
Copyright© 2017-2021, Jorge Pinal N.

Assets > PIDI Game Development Framework > Planar Reflections 4 > Legacy Rendering Pipeline > Shaders > PBR
PIDIPlanarReflections4_PBR_Metallic
PIDIPlanarReflections4_PBR_Metallic
PIDIPlanarReflections4_PBR_Metallic_Alpha
PIDIPlanarReflections4_PBR_Metallic_Alpha
PIDIPlanarReflections4_PBR_MixedReflections
PIDIPlanarReflections4_PBR_Specular

Then, as the last step, assign your Reflection Renderer to the **Cast From Renderer** slot and enable the Reflection Color feature in order to display the Reflection Color texture over the surface.


Planar Reflection Caster (Script)
PIDI: PLANAR REFLECTIONS™ 4 v4.0.0
Cast From Renderer Reflection Renderer (Planar Reflection Renderer)
PIDIPlanarReflections4_PBR_Metallic
Reflection Color Enabled
Reflection Depth Disabled
Blur Pass Disabled
Copyright© 2017-2021, Jorge Pinal N.

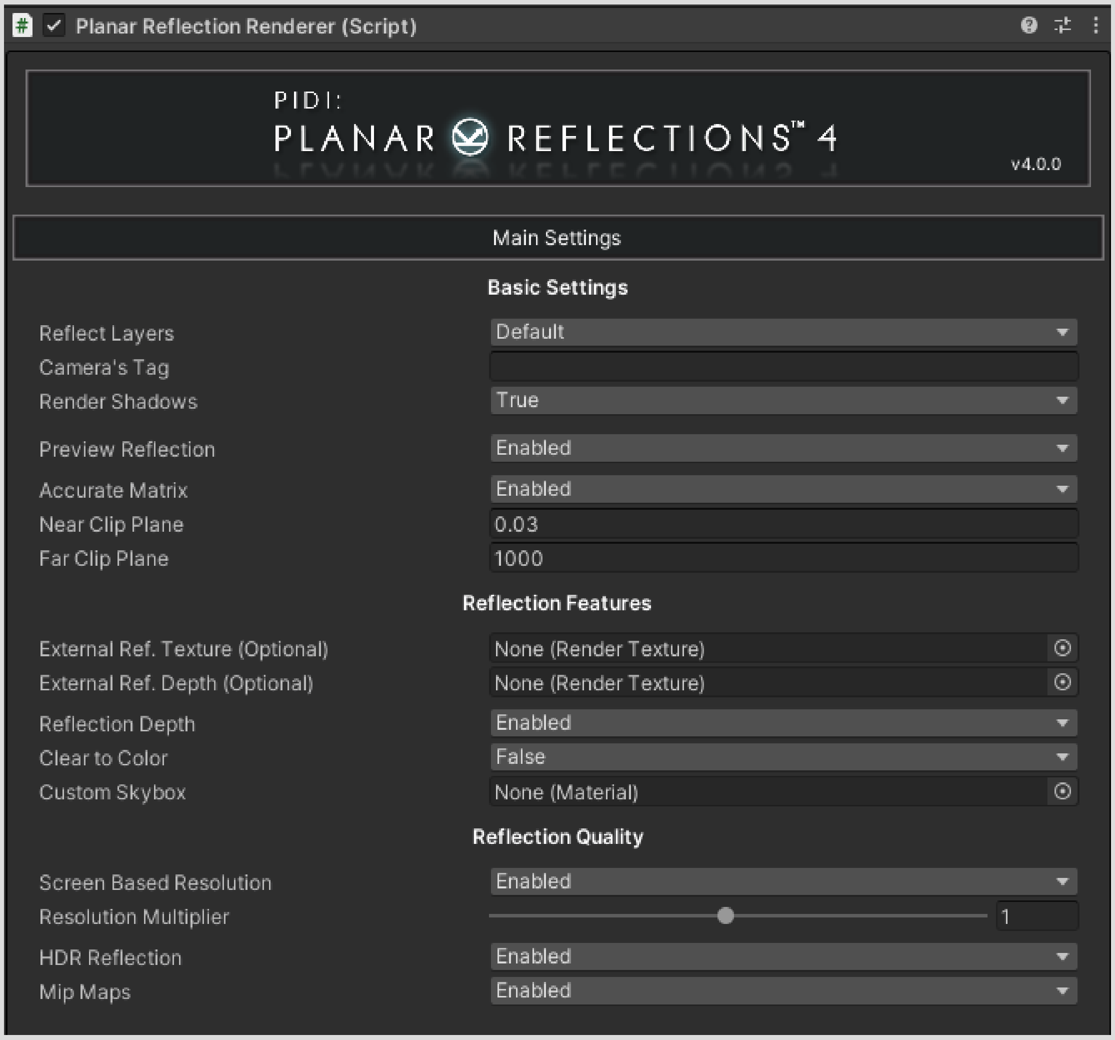
With this, the reflection is ready and will be displayed without issues in both the Game and Scene views. With Planar Reflections 4 you can also use our accurate roughness / smoothness based blur out of the box, integrated within the shader itself, without needing to enable the Blur Pass in the **Reflection Caster** component and with better performance.



Planar Reflection Renderer

The Planar Reflection Renderer (previously, Planar **Reflections** Renderer) is the script responsible for rendering the real-time reflections and their corresponding depth pass. Using considerably faster methods than in version 3.x and adding several new features both in terms of quality and coherence between rendering pipelines.

Main Settings



Basic Settings

Reflect Layers	Default
Camera's Tag	
Render Shadows	True
Preview Reflection	Enabled
Accurate Matrix	Enabled
Near Clip Plane	0.03
Far Clip Plane	1000

Reflection Features

External Ref. Texture (Optional)	None (Render Texture)
External Ref. Depth (Optional)	None (Render Texture)
Reflection Depth	Enabled
Clear to Color	False
Custom Skybox	None (Material)

Reflection Quality

Screen Based Resolution	Enabled
Resolution Multiplier	1
HDR Reflection	Enabled
Mip Maps	Enabled

The main settings for the reflection renderer are for the most part self-explanatory. We have a **Reflect Layers** LayerMask value to define which layers will be rendered into the reflections and which ones will be ignored, a **Camera's Tag** which can be used to filter which cameras will trigger the generation of reflections with all the others given an empty texture instead.

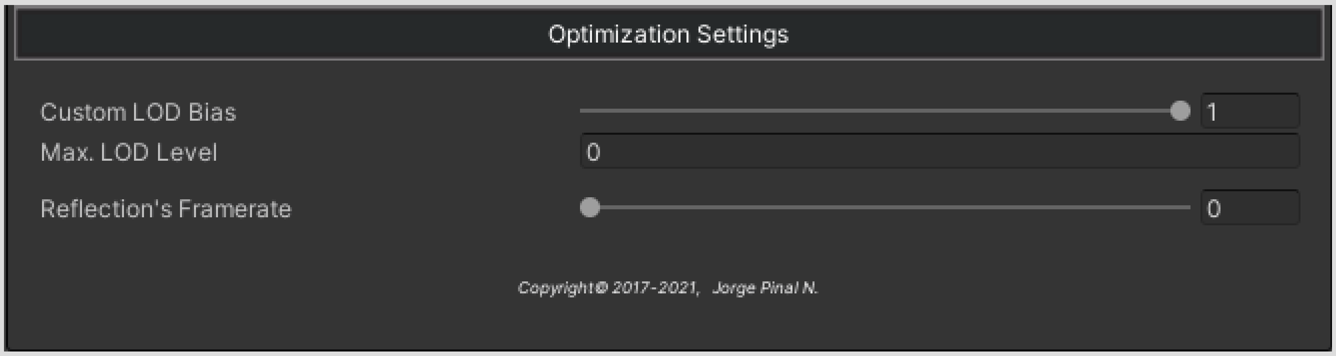
We have an override to determine whether the reflections will **Render Shadows**, whether a **Preview Reflection** will be shown in the scene view and some basic control over the reflection's projection matrix and clipping planes. You should use the **Accurate Matrix** projection in most cases to avoid the reflection from showing things that are behind / under it, but you may need to disable it in order to show some view / projection dependent post process effects accurately.

External RenderTexture assets can be assigned to the Reflection Renderer to store the reflection's color and depth but if your game uses multiple cameras in the Universal or High Definition Pipelines you should avoid this and let the Reflection Renderer to use its own internal textures instead as otherwise the reflections of one camera may overwrite the reflections of the previous one, since both of them will be writing to the same external RenderTexture.

Additional features for the rendering can be enabled here as well, such as whether the reflection will render an additional **Depth Pass**, if it will use **PostFX**, whether to clear to a solid color or not and even a custom skybox to override the scene's default one

Finally, you can easily adjust the quality of the reflection including its resolution and downscale, whether the resolution should be absolute or based on the screen's resolution, as well as different parameters of the resulting reflection texture's format such as HDR support and / or MipMaps (**highly recommended**).

Optimization Settings



Optimization Settings

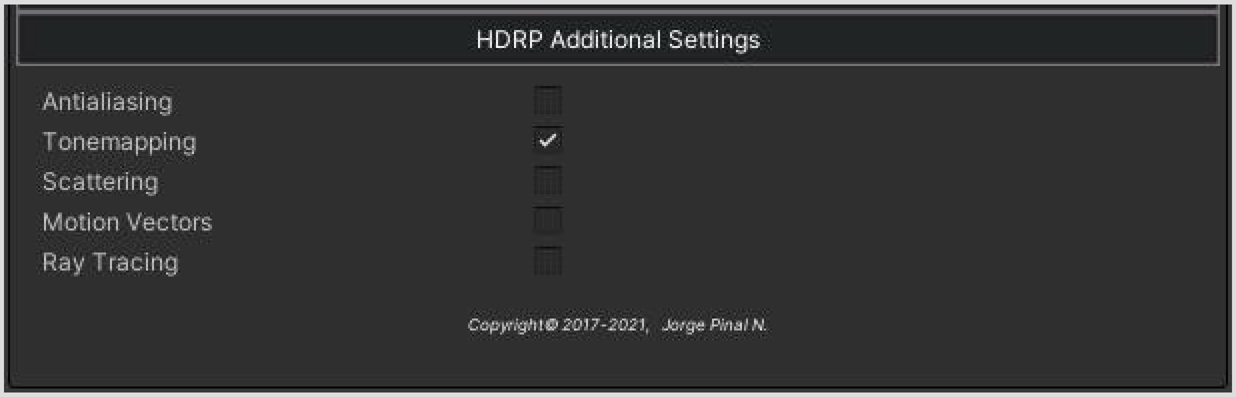
Custom LOD Bias 1
Max. LOD Level 0
Reflection's Framerate 0

Copyright© 2017-2021, Jorge Pinal N.

Different optimization parameters can be used to speed up the reflection's rendering time. While lowering the reflection's resolution can help in situations where shaders are complex or multiple full-screen effects are being used may improve the performance, in most cases the main reason for lower performance when using real-time reflections comes from a mix of draw calls (or SetPass calls) and geometry complexity.

Starting with version 4.x you can set custom LOD biases and control the maximum LOD level to be used by each reflection in order to limit considerably the visual quality of the reflections and use lower quality versions of your models for them instead of the full resolution ones. On top of this, you can also set up a lower framerate for your reflections to gain some extra FPS in your game.

HDRP Optimization Settings



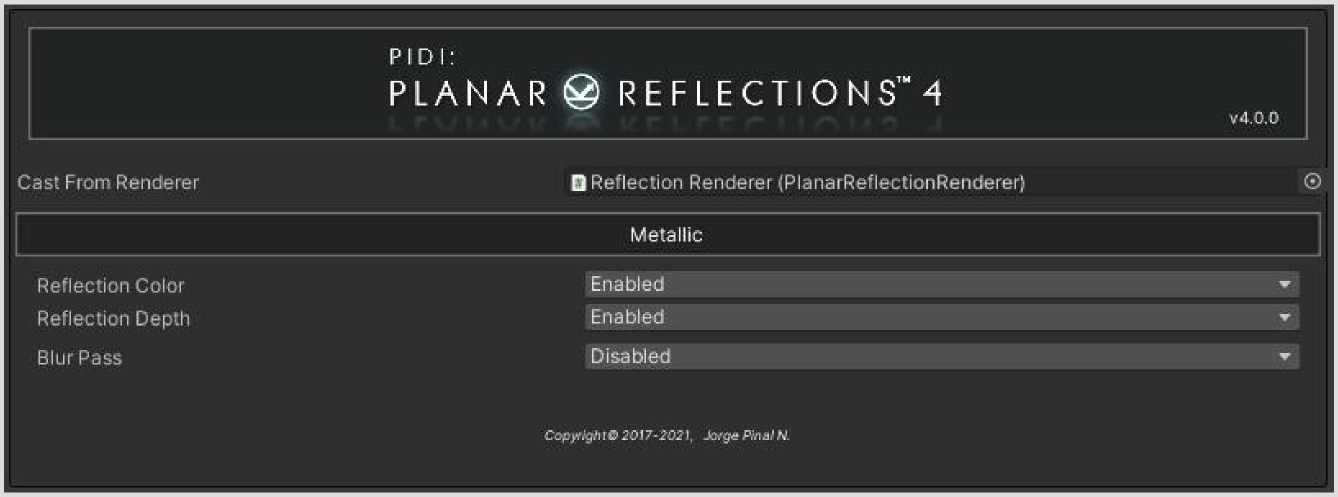
HDRP Additional Settings

Antialiasing	☐
Tonemapping	✓
Scattering	☐
Motion Vectors	☐
Ray Tracing	☐

Copyright© 2017-2021, Jorge Pinal N.

The asset has specific optimization parameters for the HD Rendering Pipeline that allow you to enable and disable advanced features that are exclusive to this pipeline. By default, most of them are disabled in order to ensure maximum performance but you can turn them on depending on the kind of effects that you need to reflect. Unlike the Planar Reflection Probes used by Unity the Planar Reflections generated by PIDI Planar Reflections 4 use a fully featured virtual camera which allows you to reflect and control additional features.

Planar Reflection Caster



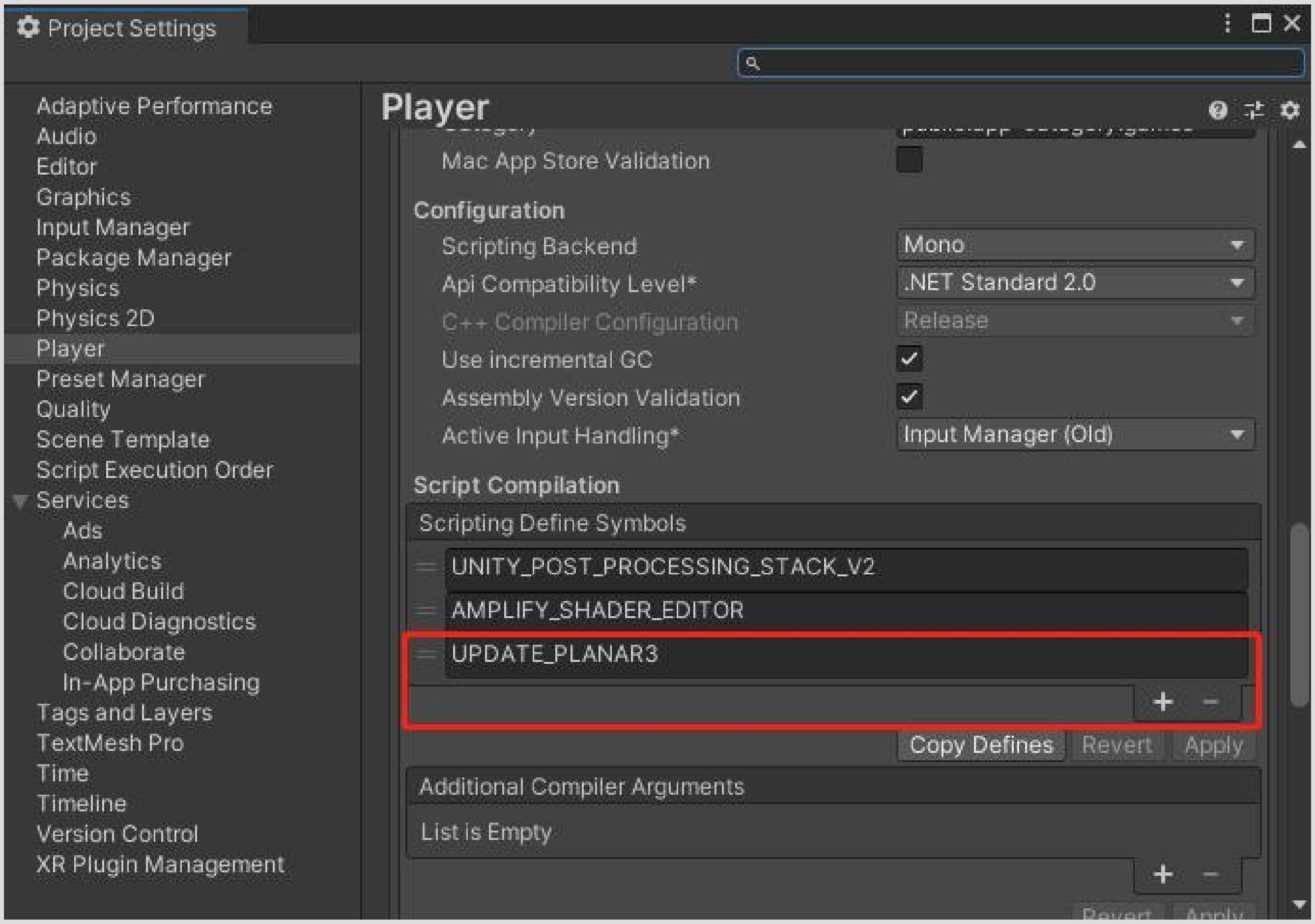
Cast From Renderer	Reflection Renderer (PlanarReflectionRenderer)
Metallic	
Reflection Color	Enabled
Reflection Depth	Enabled
Blur Pass	Disabled

The **Planar Reflection Caster** component is the script that takes a reflection and assigns it to a material. It must always be attached to a Mesh Renderer component and it will automatically track all of its materials and enable per-material settings to define whether the material uses the **Reflection Color** texture (_ReflectionTex), the **Reflection Depth** texture (_ReflectionDepth) and whether an additional Blur Pass should be used to blur the final output of the reflection.

In version 3.x of the asset this blur pass was used to simulate the roughness of the material and how it affects the sharpness of the reflection but in version 4.x all PBR shaders have been rewritten in all pipelines to use a much more accurate and performant blur method. Because of this, we recommend you to disable the blur pass in most cases, it is provided as a form of backwards compatibility with shaders from previous versions.

Upgrading from Planar Reflections 3

If you have the Standard version of PIDI Planar Reflections 3.x you can use the define symbol UPDATE_PLANAR3 in your project (if it has not been added automatically) in order to automate most of the upgrade process.



Player

Mac App Store Validation

Configuration

Scripting Backend: Mono
Api Compatibility Level*: .NET Standard 2.0
C++ Compiler Configuration: Release
Use incremental GC: ✓
Assembly Version Validation: ✓
Active Input Handling*: Input Manager (Old)

Script Compilation

Scripting Define Symbols

```
UNITY_POST_PROCESSING_STACK_V2
AMPLIFY_SHADER_EDITOR
UPDATE_PLANAR3
```

Additional Compiler Arguments

List is Empty

With this define symbol enabled you can add the new Planar Reflection Renderer and Planar Reflection Caster components next to their counterparts from version 3.x and they will automatically sync all properties and corresponding settings to greatly reduce the amount of time you need to spend setting up your scenes for version 4.x.

You can watch the video here to see a breakdown of how to update from PIDI Planar Reflections 3.x to version 4.x

Warning : After you remove PIDI Planar Reflections 3 from your project you may see several errors pop up in the console. This is because the UPDATE_PLANAR3 define is still in place, making Unity believe that the asset is still there. You need to remove this define manually and this will make all the errors go away.

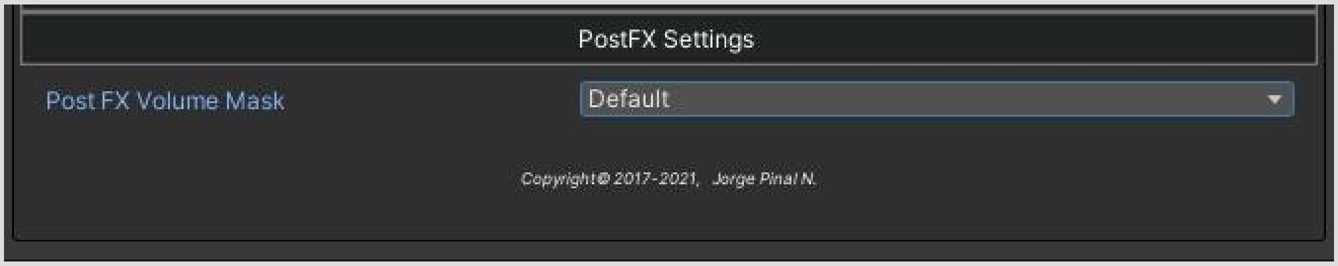
Advanced Topics

In this section we will cover some advanced uses of the PIDI Planar Reflections 4.x asset, including how to create custom shaders that support the reflections generated by the tool using ShaderLab, Amplify Shader Editor and ShaderGraph as well as the specific utilities provided for each of them.

We will also go over some optimization tips when using reflections on your scenes, what is a depth pass and a blur pass and how you can integrate them into your own shaders.

Post Process FX

Each Reflection Renderer component can set up support for Post Process FX with their unique volume layer masks. While most Post Process FX will work without issues with the Reflection Renderer (including ambient occlusion, color grading, bloom, depth of field etc) some of them, especially those that depend on motion vectors or screen based UVs (such as vignetting) may not work nor look correctly.


PostFX Settings

Post FX Volume Mask: Default

Copyright© 2017-2021, Jorge Pinal N.

Once Post Process FX are enabled in the Planar Reflection Renderer you can set up with which layer is it going to interact. It is not recommended to use the same post processing volume for your main in-game camera and the reflections as this could cause some overlapping of effects, for example if you apply color grading to your scene you may end up applying color grading to the reflection and then a second pass of the same color grading through the main camera. This applies as well to Tonemapping effects, being useful in some cases to use a HDR enabled reflection instead and let the main camera's tonemapping and bloom filters take care of the rest.

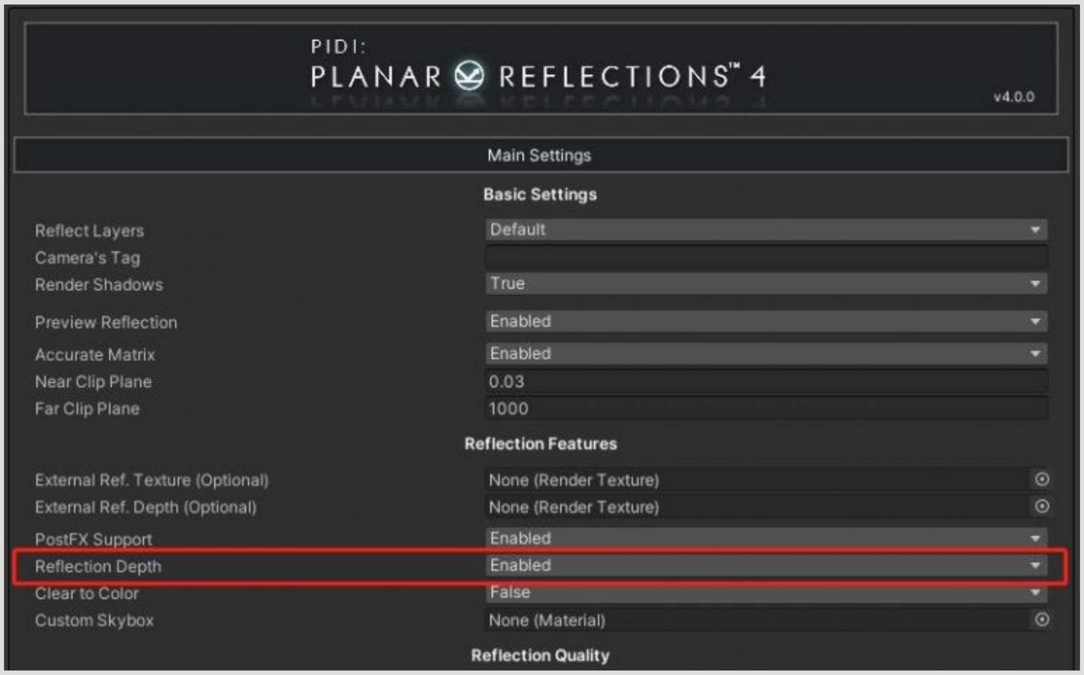
In HDRP you should enable Post Process FX in most cases as they are necessary to handle the high intensity values of lights and adjust the exposure of the scene accordingly. Both in Universal RP and in HDRP the use of Post Processing will add some overhead that does not exist in the Legacy Rendering Pipeline due to the different way that it is handled on each rendering pipeline.

Depth & Blur Passes

Depth Pass

Each Planar Reflection Renderer component can have its own optional depth pass. This depth pass stores the distance between the mirror's surface and the objects reflected on it based on the depth of the virtual camera used to render it. This depth pass can be used to create complex fading effects or, in the case of the included PBR based shaders, to simulate contact reflections (that is, reflections that are sharper the closer an object gets to the reflective surface).

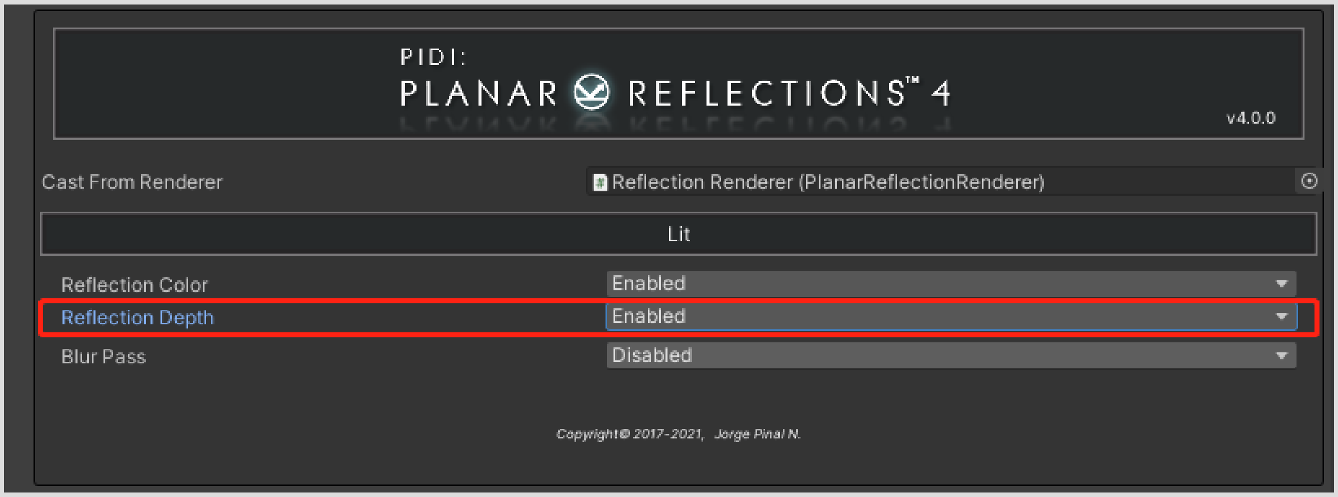
There are two steps to enable depth in your reflections. First, you must enable the depth pass in the Reflection Renderer to ensure that the pass is generated at all:



Reflect Layers	Default
Camera's Tag	
Render Shadows	True
Preview Reflection	Enabled
Accurate Matrix	Enabled
Near Clip Plane	0.03
Far Clip Plane	1000

External Ref. Texture (Optional)	None (Render Texture)
External Ref. Depth (Optional)	None (Render Texture)
PostFX Support	Enabled
Reflection Depth	Enabled
Clear to Color	False
Custom Skybox	None (Material)

Then you have to enable the depth pass in your Reflection Caster for every material that will use it, so that the corresponding texture is sent to the material.

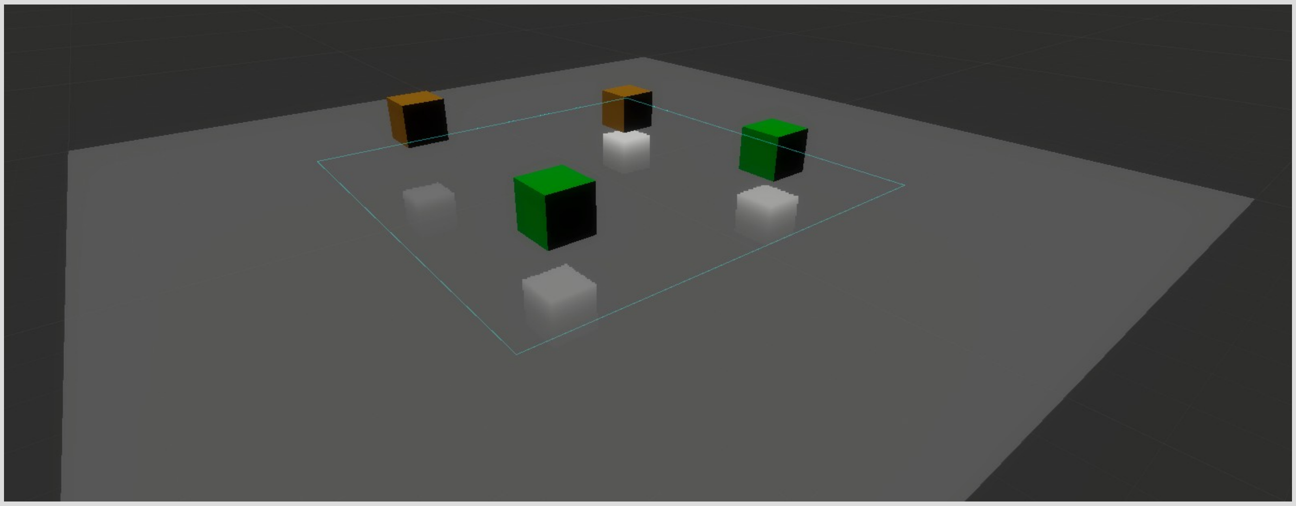


Cast From Renderer	Reflection Renderer (PlanarReflectionRenderer)
Reflection Color	Enabled
Reflection Depth	Enabled
Blur Pass	Disabled

Once you've done this, your shaders can access the property called "_ReflectionDepth" and use its Red Channel, which now contains a default depth texture, and use it for all sorts of effects.

Please remember that not all platforms support depth textures and that some pipelines (like Universal RP) need to have the depth rendering feature enabled from their general settings.

For shader programming purposes, when depth is disabled in either the Reflection Renderer or the Reflection Caster it is set to black by default. The included nodes for Amplify Shader Editor and ShaderGraph provide useful outputs for the reflection's depth that are ready to use.

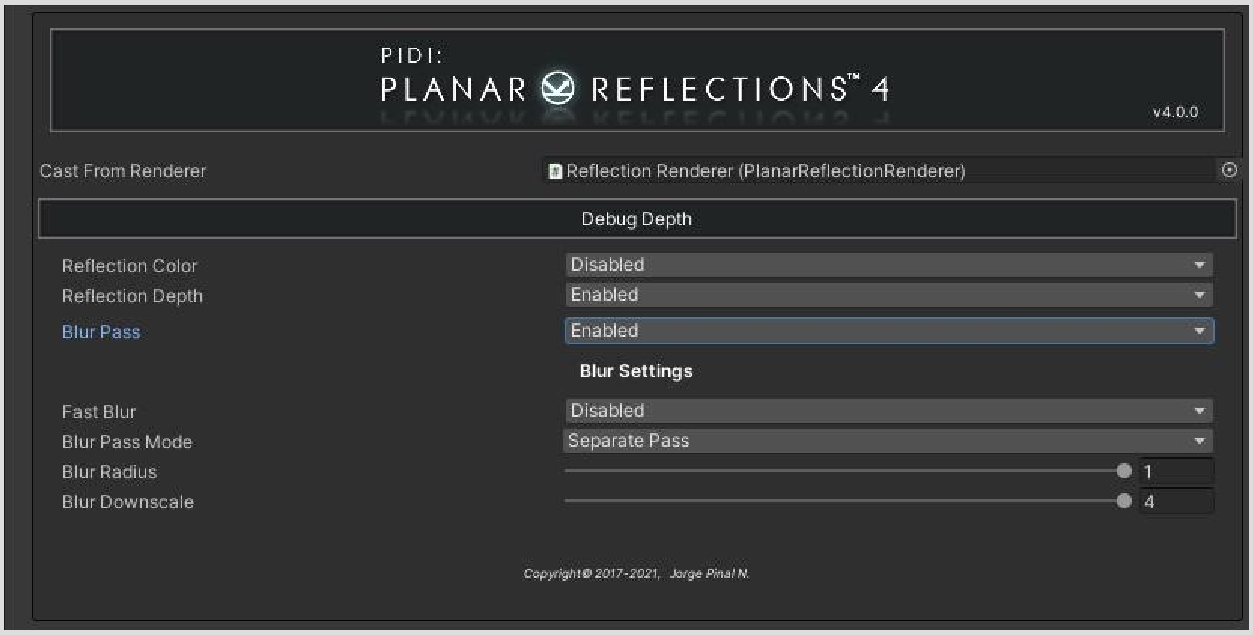


Blur Pass

In PIDI Planar Reflections v 3.x an additional Blur Pass was available in the Reflection Caster component. This blur pass would apply a configurable blur to the reflection's color and use this blurred texture to approximate how the roughness of a material affects its sharpness. It could also be used to simply blur the final output reflection for additional effects such as a foggy mirror or as a cheap alternative to anti-aliasing.

In version 4.x this is no longer necessary as all PBR shaders have a new and much more efficient integrated blur algorithm built into their code that accurately uses the roughness of the material to blur the reflection and achieve a more believable result. However, the old blur pass offered in v3.x is still available both as a backwards compatibility feature (for shaders made for version 3.x) as well as an artistic choice for some effects that may still benefit from it.

To use it, simply go to the Planar Reflection Caster component and enable the Blur Pass for the corresponding materials. Then you will see the following options appear:



Cast From Renderer: Reflection Renderer (PlanarReflectionRenderer)

Debug Depth

Reflection Color	Disabled
Reflection Depth	Enabled
Blur Pass	Enabled

Blur Settings

Fast Blur	Disabled
Blur Pass Mode	Separate Pass
Blur Radius	1
Blur Downscale	4

Copyright© 2017-2021, Jorge Pinal N.

Fast Blur is a setting that simplifies the internal algorithm to gain a few FPS in older devices. The Blur Pass mode defines whether the final reflection sent to the material or a copy of it will be blurred (if this setting is set to Separate Pass then the blurred reflection will be assigned to the **_BlurReflectionTex** property). Finally, for each material you can adjust the blur radius and the downscale that will be applied to the reflection texture before blurring it, giving you full control over the effect on a per-material basis.

Legacy RP Shaders

While several different shaders are provided for the Legacy Rendering Pipeline that cover a wide variety of uses including PBR materials, water shaders and basic mirror-like shaders, there may be a need to create your own for the specific project you are working on.

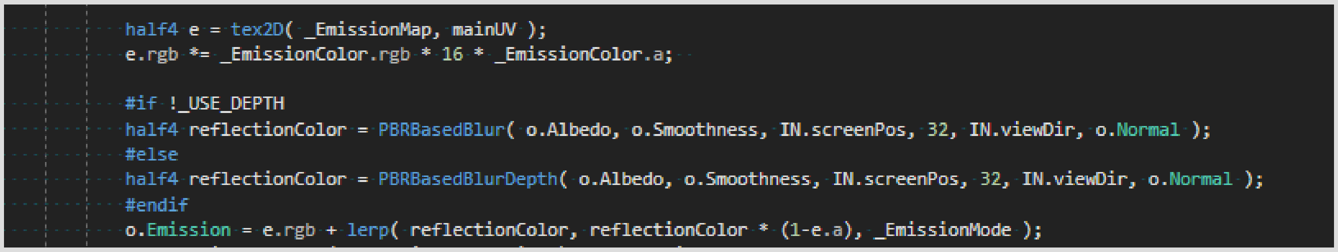
A CGInclude file is provided with the asset that contains several helpful functions that should simplify the integration of the asset with custom made shaders. Helpers for getting the reflection UVs as well as for generating a fully PBR adjusted reflection color (with and without contact depth) are provided.

half2 screen2ReflectionUVs(float4 screenPosition)

This function turns the screen position (in Unity shaders IN.screenPosition) into a set of usable UVs to unwrap the reflection data (both color and depth).

half4 PBRBasedBlur(half4 albedo, half smoothness, half4 screenPosition, half maxBlur, half3 viewDir, half3 normal) and PBRBasedBlurDepth

This function takes several properties from the PBR shader (the same that will be sent to the corresponding PBR outputs) such as albedo, smoothness and normal, as well as the reflectionUVs, a maxBlur value (to define how much will the reflection blur on very rough surfaces) and a view direction to apply a simple fresnel effect. It can be used as follows:



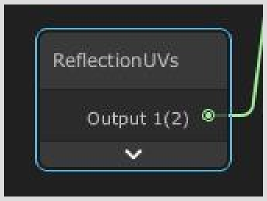
```
            half4 e = tex2D( _EmissionMap, mainUV );
            e.rgb *= _EmissionColor.rgb * 16 * _EmissionColor.a;

            #if !_USE_DEPTH
            half4 reflectionColor = PBRBasedBlur( o.Albedo, o.Smoothness, IN.screenPos, 32, IN.viewDir, o.Normal );
            #else
            half4 reflectionColor = PBRBasedBlurDepth( o.Albedo, o.Smoothness, IN.screenPos, 32, IN.viewDir, o.Normal );
            #endif
            o.Emission = e.rgb + lerp( reflectionColor, reflectionColor * (1-e.a), _EmissionMode );
```

This is an example of how to add accurate reflections to a PBR shader using the Specular workflow in the Legacy RP (it is based around the Specular shader included with the asset). Using the functions included in the CGInclude file as well as a shader_feature called USE_DEPTH to define whether a depth pass will be used or not (which can save some performance at the shader level by reducing the amount of texture operations).

ShaderGraph Shaders

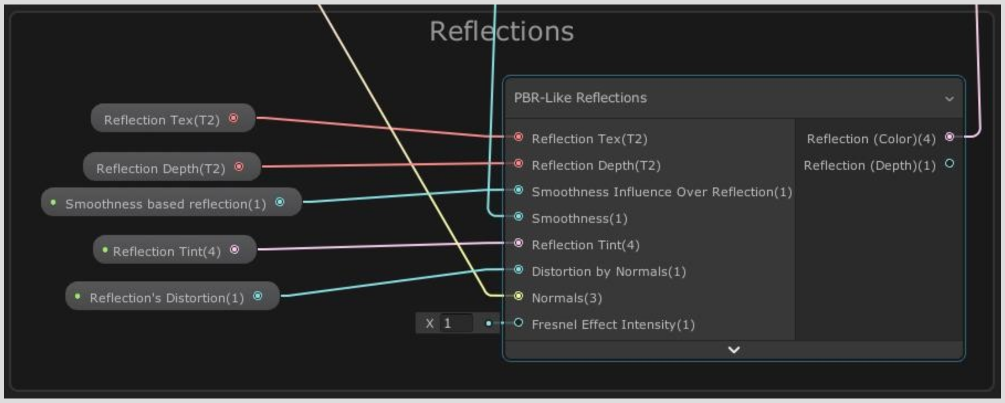
In both Universal RP and HDRP you can easily create custom shaders that support real-time reflections by using the included ShaderGraph sub-graphs that handle reflection UVs (for simple reflections) and full PBR based workflows. These graphs can be edited and extended as you see fit in order to adapt to your project's purposes.

ReflectionUVs node



This node automatically generates a Vector2 type output containing ready to use UVs to use on either the Reflection color or Reflection depth textures.

PBR Like Reflections node



The PBR-Like reflection node generates a PBR based reflection that takes into account the smoothness of the material, its normals and a simple fresnel effect.

It takes as inputs the Reflection Tex and Reflection Depth textures, a value to define how much the smoothness of the material will affect the sharpness and intensity of the reflection, the smoothness of the material as a float value, a color for the reflection's final tint, a value controlling how much the normals of the material will distort the reflection and a vector3 containing the normal data of the material (the same value that will be sent to the Normal output of the shader), and finally a value specifying the intensity of the fresnel effect.

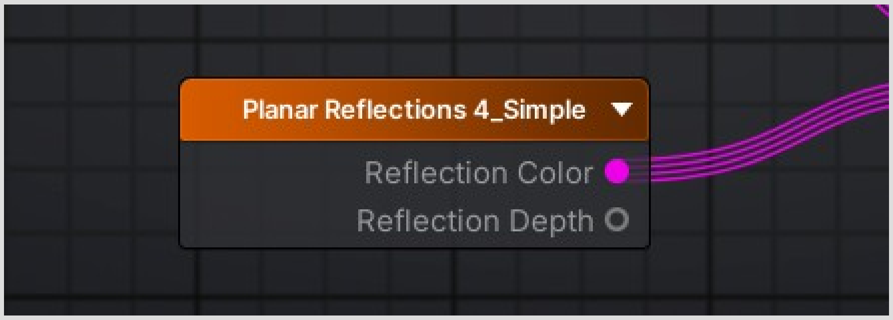
This sub-graph has all the necessary operations inside to unwrap, process, blur and output both the reflection color and depth in a ready to use way that will greatly simplify the process of integrating PIDI Planar Reflections 4.x in your own ShaderGraph based shaders, regardless of if they are designed for Universal RP or HDRP.

Using Amplify Shader Editor

For users that create their shaders using Amplify Shader Editor we also provide a few custom nodes that will make the integration process with PIDI Planar Reflections 4.x much easier. On top of the Legacy reflections node from PIDI Planar Reflections 3.x (provided for backwards compatibility) we also include two new nodes, one for simple reflections and one for PBR based ones.

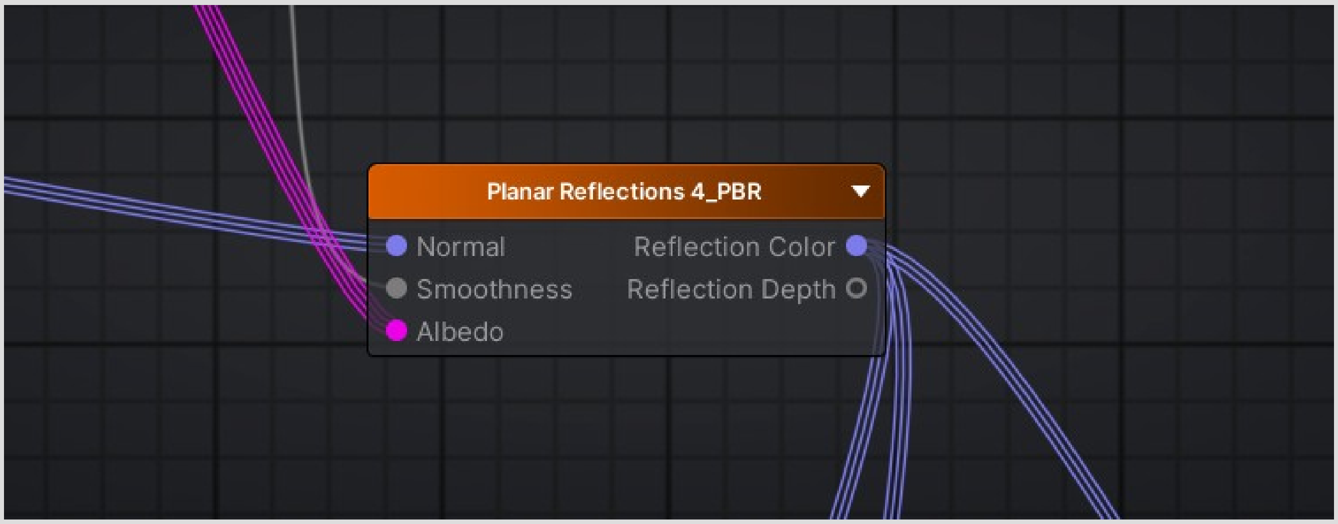
To use these nodes in your project, first make sure that Amplify Shader Editor is installed in your project. Then go to the Shared Assets folder included with the PIDI Planar Reflections 4.x asset and unpack the Unity package included in the Amplify Shader Editor Nodes folder.

PlanarReflections4_Simple



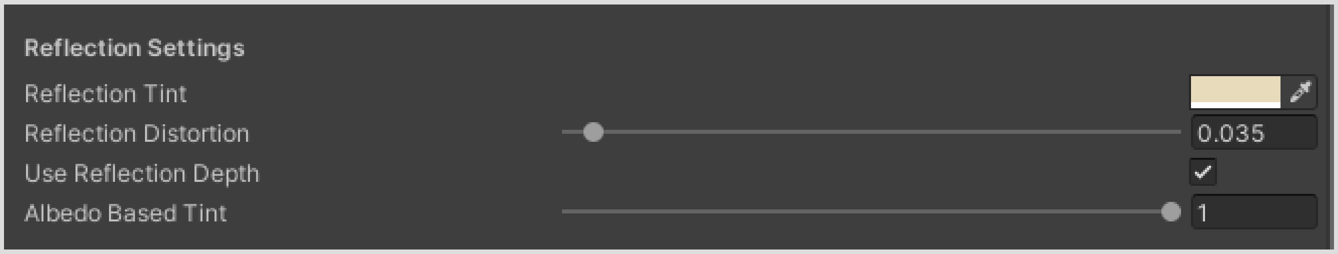
This node allows you to add simple reflection color and depth to your shaders without any needed extra inputs nor setup. Simply drop into an Amplify based shader and use it in any way you see fit. **The depth output is not processed in any way nor adjusted for eye / camera based depth, it is the value in its raw format.**

PlanarReflections4_PBR



This node allows you to add PBR based reflections to your Amplify made shaders through an easy to use node that only takes the normal, smoothness and albedo values from your material and uses them to generate the appropriate Reflection Color and depth outputs. The Depth in this case has been processed and is ready to use for contact depth and other similar effects (which are also automatically handled for the reflection color).

The node also adds automatically all the parameters needed to control the reflection to your shader, using appropriate headers and spacing to display them in the UI.


Reflection Settings
Reflection Tint
Reflection Distortion 0.035
Use Reflection Depth ✓
Albedo Based Tint 1

Optimization Tips

Rendering a reflection in real-time involves in most cases re-rendering your scene once for every single camera that views the reflection and for every reflection in the scene. This can cause performance to drop significantly if too many reflections are added to a scene and many of them are visible at once.

While the actual cost of rendering the scene for a reflection is usually smaller than the cost of rendering the scene through the main camera there are some additional steps that you can take to improve performance even more.

Control Shadows & LOD

Use a lower LOD bias for your Reflection Renderer so that the reflections use lower quality models, reducing their performance impact. You can also change the Max. LOD value to ensure that only low poly versions of your models are reflected since, in most cases, players will not notice these smaller differences.

Additionally, you can disable shadows in the reflection unless completely necessary to reduce the amount of draw calls generated by every mirror in scene.

Lower the reflection’s resolution

If your game uses heavy shaders and post-process FX or has a lot of overdraw then reducing the resolution of the reflections may be helpful. Using a slight blur pass on them may also reduce the sharp edges and make them appear softer and more natural. On top of this, you can also set the reflection’s resolution to depend on the main screen’s resolution so that players using older devices and switching to lower resolutions get lower quality reflections automatically.

Limit the reflection’s framerate

Lastly, you can limit the reflection’s framerate to gain some additional performance. This however is not recommended unless your game runs at a very high framerate by default as with a lower framerate the reflections may seem too choppy producing an unpleasant effect in the scene.

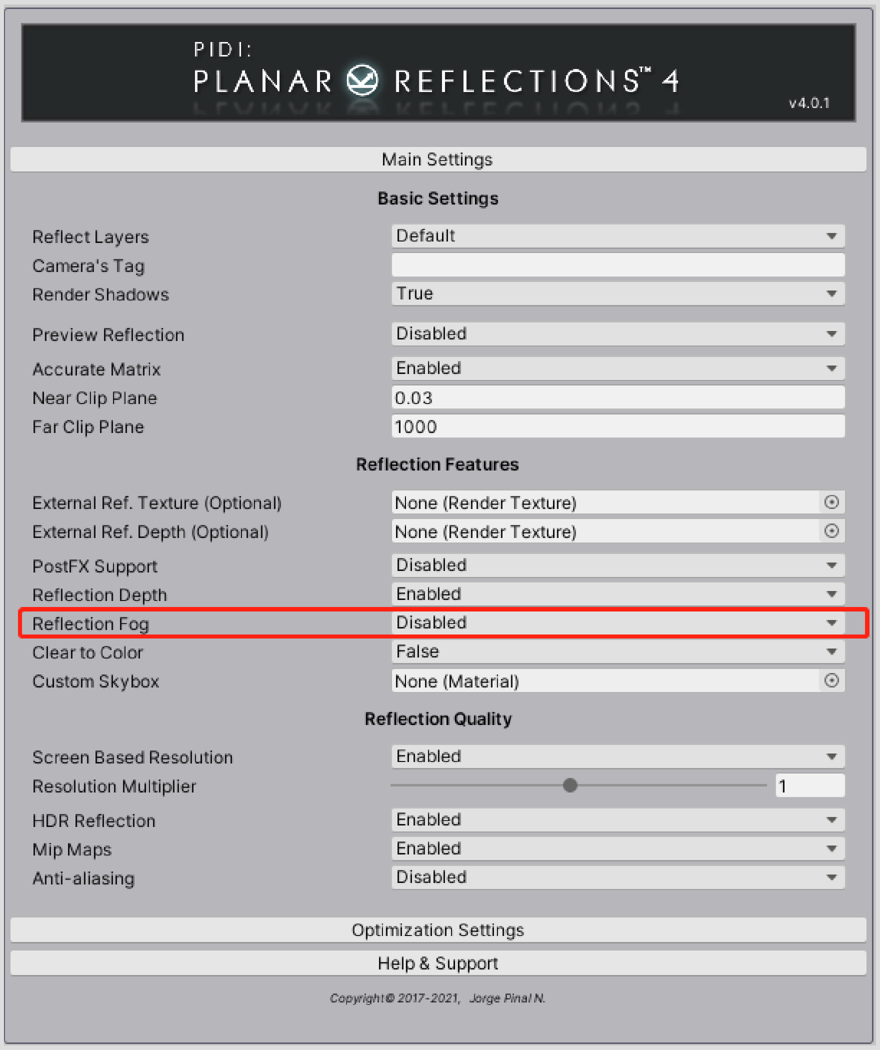
Beta - Using fog with Planar Reflections

Fog in Unity doesn't always work in the way that is expected when used together with Planar Reflections. The main reason for this is the way fog is calculated using the camera's clip space, which in Planar Reflections is heavily modified due to the use of an oblique projection.

In HDRP and Legacy Deferred mode + Post FX fog is reconstructed or calculated from the start based on world positions, which means that it works mostly without any issues in the reflections. But for any other cases and for URP this is not possible, which means that fog has to be calculated and added back into the reflections in a different way.

As a solution for this problem, PIDI Planar Reflections 4 includes a brand new Fog Pass that generates an additional texture including world position based fog as seen through the reflection. This fog reads and matches all parameters set up in the Lighting settings of the scene by default, no additional work required.

Enabling fog support in the Legacy pipeline is as easy as turning on the setting in the Reflection Renderer and Reflection Caster scripts.


PIDI:
PLANAR REFLECTIONS™ 4
v4.0.1

Main Settings

Basic Settings

Reflect Layers	Default
Camera's Tag	
Render Shadows	True
Preview Reflection	Disabled
Accurate Matrix	Enabled
Near Clip Plane	0.03
Far Clip Plane	1000

Reflection Features

External Ref. Texture (Optional)	None (Render Texture)
External Ref. Depth (Optional)	None (Render Texture)
PostFX Support	Disabled
Reflection Depth	Enabled
Reflection Fog	Disabled
Clear to Color	False
Custom Skybox	None (Material)

Reflection Quality

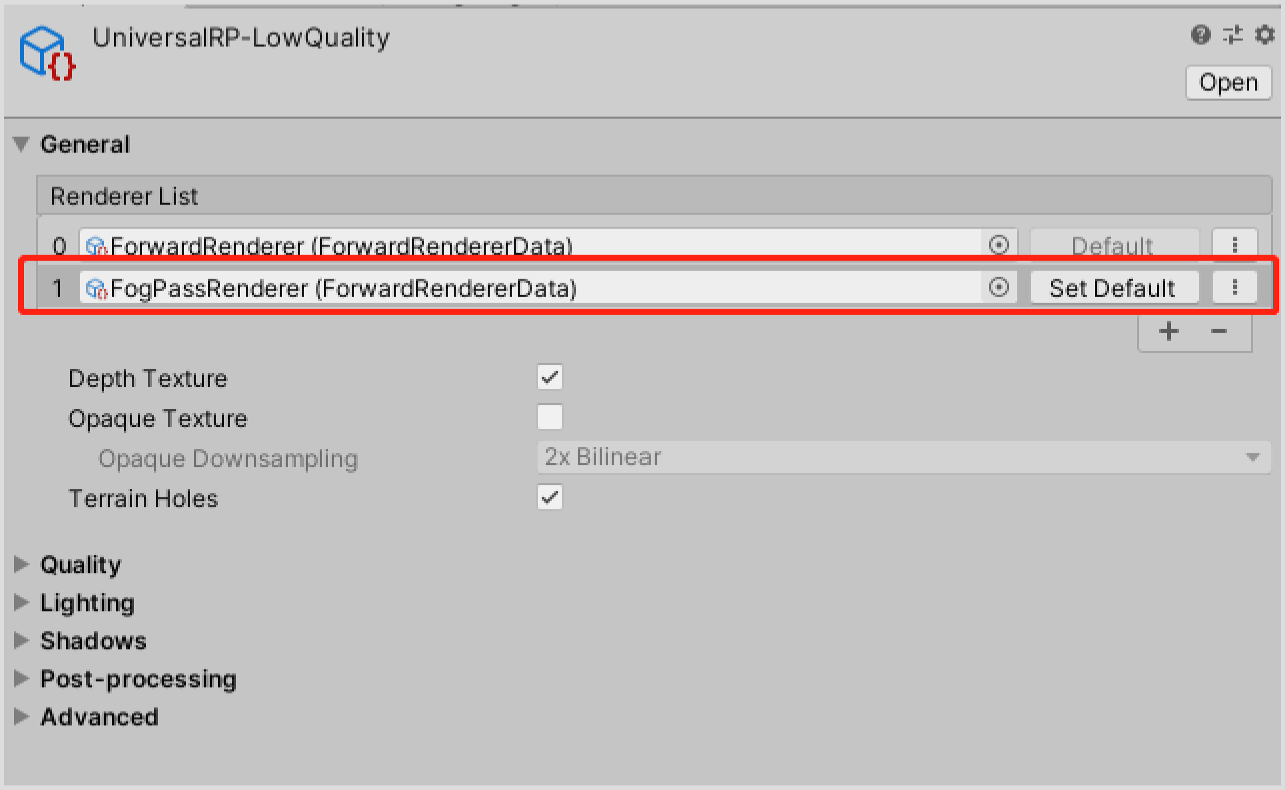
Screen Based Resolution	Enabled
Resolution Multiplier	1
HDR Reflection	Enabled
Mip Maps	Enabled
Anti-aliasing	Disabled

Optimization Settings

Help & Support

Copyright© 2017-2021, Jorge Pinal N.

In URP however, due to limitations in the rendering pipeline, a special renderer is required and has to be added to the Scriptable Render Pipeline asset that your project uses. You can find this renderer inside the Fog Pass folder included with the asset.



UniversalRP-LowQuality

Open

General

Renderer List

0 ForwardRenderer (ForwardRendererData) Default

1 FogPassRenderer (ForwardRendererData) Set Default

Depth Texture	✓
Opaque Texture	
Opaque Downsampling	2x Bilinear
Terrain Holes	✓

Quality

Lighting

Shadows

Post-processing

Advanced

As this is a Beta feature there are some limitations to what this fog renderer can do. The main limitation is that all objects in the fog pass are rendered as opaque objects.

Final Notes

PIDI Planar Reflections 4 is NOT designed to work with any VR or AR devices. While the reflections produced by this asset have been optimized and have a very high performance the final results you will get for your project will depend entirely in the complexity of your scenes, since the scene will be rendered at least once more for each reflection. Reducing draw calls and simplifying the geometry of your levels is the first and most important step to ensure an optimal framerate.

Make sure that the device you are targeting is fully compatible with Render Texture features since these are required for the asset to work as intended. If you are working on mobile devices make sure that the Depth Texture features of the cameras / rendering pipelines are enabled to ensure that all effects work as expected. Disabling depth rendering from your cameras will prevent the water shaders from working, except those explicitly marked as “No Depth” or “Simple”.

If you have any doubt about this product or how to use it, please contact us at support@irreverent-software.com and we will get back to you to work and solve your doubts as soon as possible.

Thank you very much for purchasing this asset from PIDI – Game Development Framework. I hope that it will help you make amazing games!