
PIDI
GAME DEVELOPMENT FRAMEWORK ™
BY IRREVERENT SOFTWARE™

PIDI

GAME DEVELOPMENT FRAMEWORK ™

BY IRREVERENT SOFTWARE™

PIDI 2D REFLECTIONS

USER MANUAL

Index

PIDI – 2D Reflections
Introduction

PIDI – 2D Reflections is an advanced solution for all the reflection needs in your 2D game. From basic mirror-like reflections in floors and walls to distorted water like surfaces and from drag and drop shader based reflective surfaces to full on local reflections systems, everything is possible with this tool.

Unlike other 2D reflection solutions for Unity, our solution provides workflows entirely based on shaders that "just work" upon being dropped on the scene as well as an advanced mode which gives full control over the final output of the reflections and the layers it reflects, down-scaling, etc.

On top of this, this package also provides a basic workflow for local reflections generated specifically for a single object and used to allow it to step over another reflective surface without being occluded (p. eg. a main character stepping over water).

On top of this, starting with version 1.2 we also provide support for the new Lightweight SRP available on Unity 2018.1 and above so you can push your games to the next level. All Lightweight SRP shaders are fully compatible and editable in the Amplify Shader Editor, by Amplify Creations.

We highly recommend you this product to author your own, PIDI 2D Reflections compatible shaders.

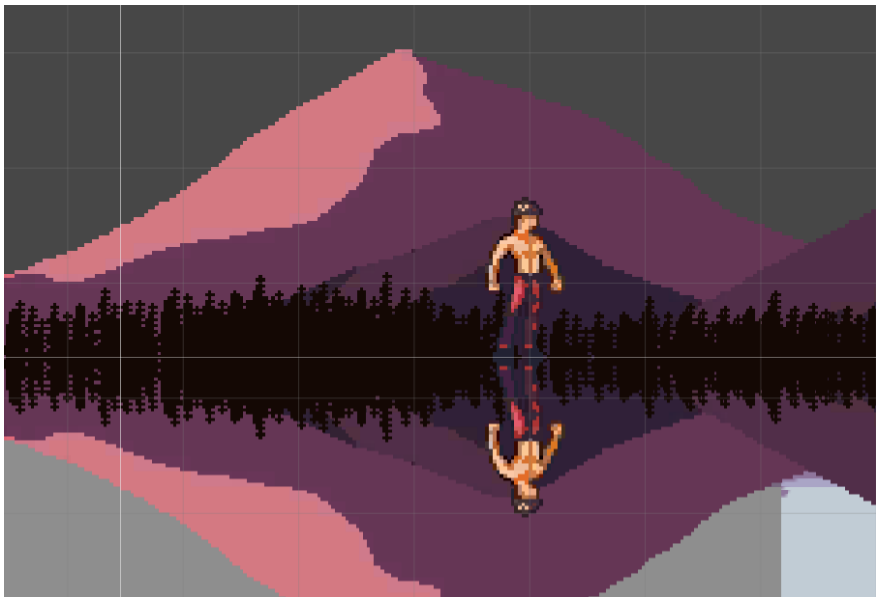
In this manual we will cover all the basics so you can get started and using 2D reflections for your games in no time.

We thank you very much for purchasing our product and hope that PIDI – 2D Reflections will help you keep making awesome projects.

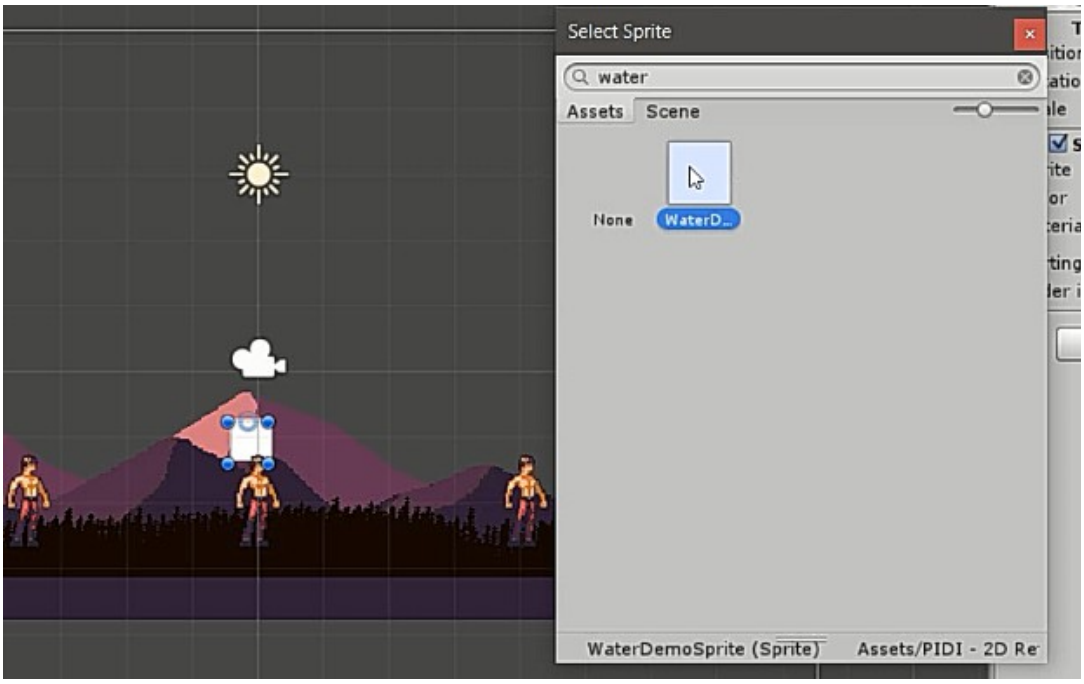
If you have any questions, suggestions or need support, contact us at support@irreverent-software.com

1. Adding 2D Reflections to your scene

1- Shader Based Method

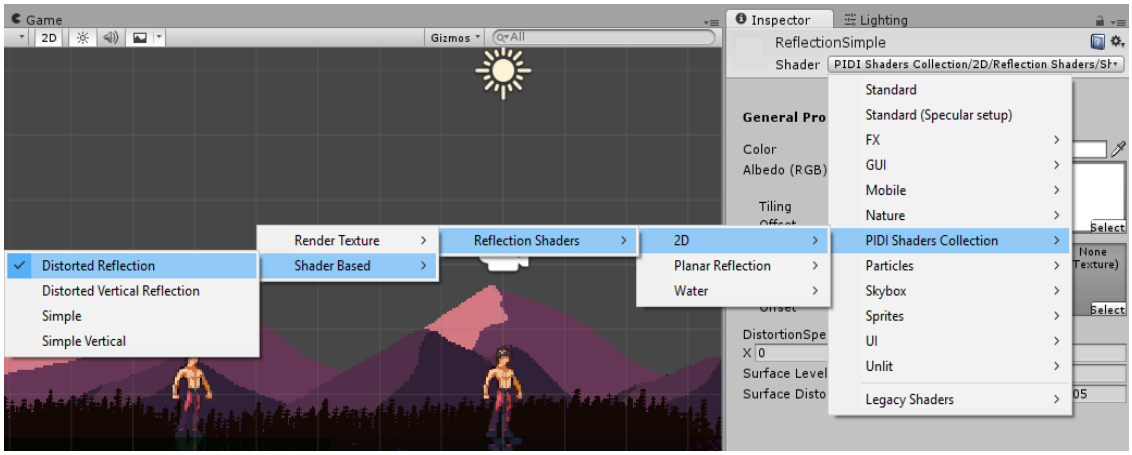


Adding reflections to any scene at any point in the development of your game is very easy when you use any of the included shaders from the “Shader Based” branch. They can be used on sprites ONLY.

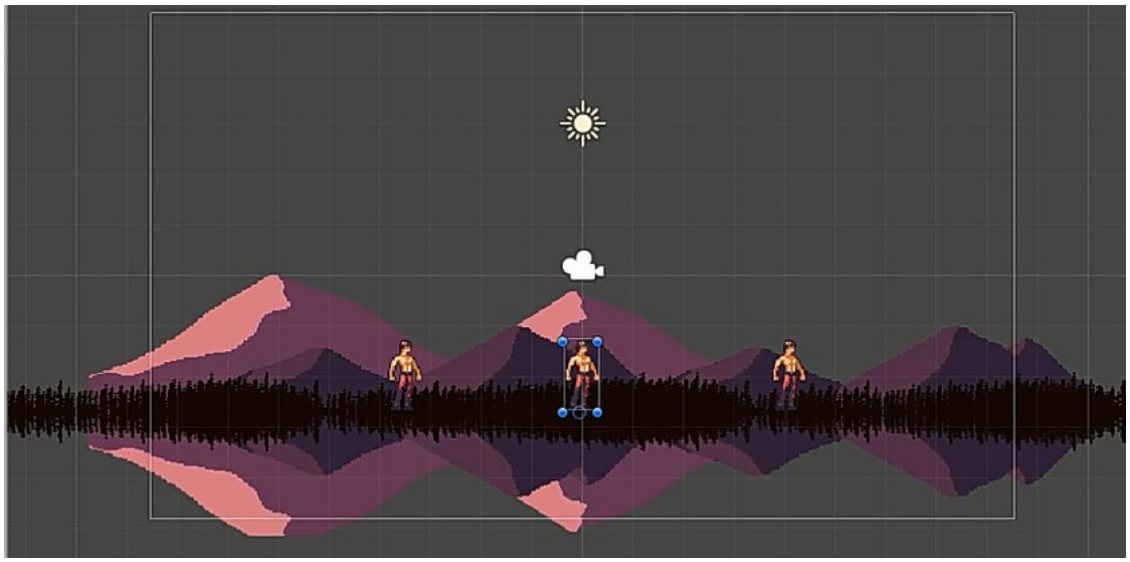


Let's start with the basics. In any scene, create a sprite object and assign to it the provided “Water” sprite or any other sprite with the anchor point located on the top center position (or the position where the reflective surface's edge will be located).

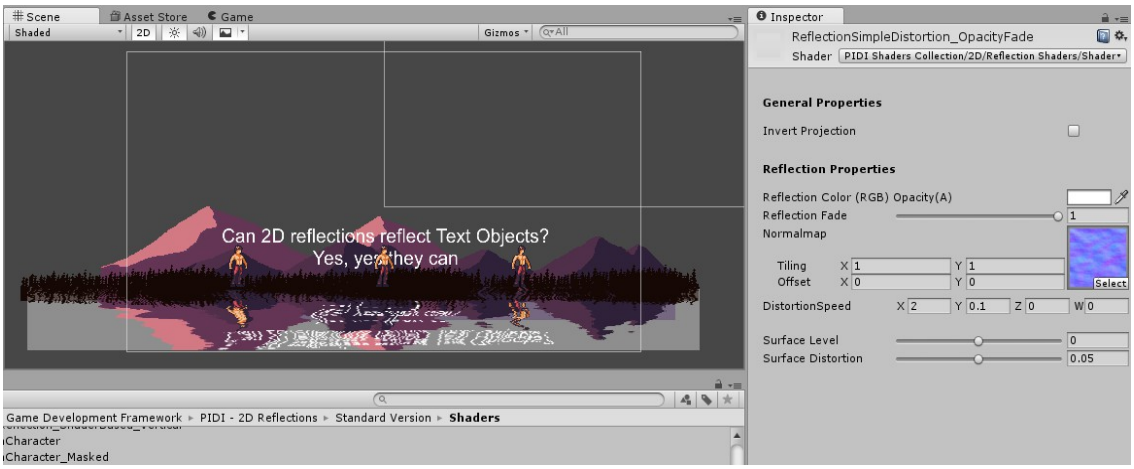
Ensure to assign it to a higher drawing order in the layer than all the other objects you want to be reflected. Several reflections using this method and the same material should be placed in different sort orders in order to make them work at the same time.



Once the sprite is in place, create a custom material with a shader from the Shader Based branch. In the example above, we used the Distorted Reflection shader. Assign this shader to the sprite, and the reflection will appear immediately.



From here you can play with the different values on the shader's interface, such as the distortion maps, distortion strength and speed (used for water-like effects) and the surface level. The surface level allows you to further adjust the distance between the reflection and the edge of the surface.

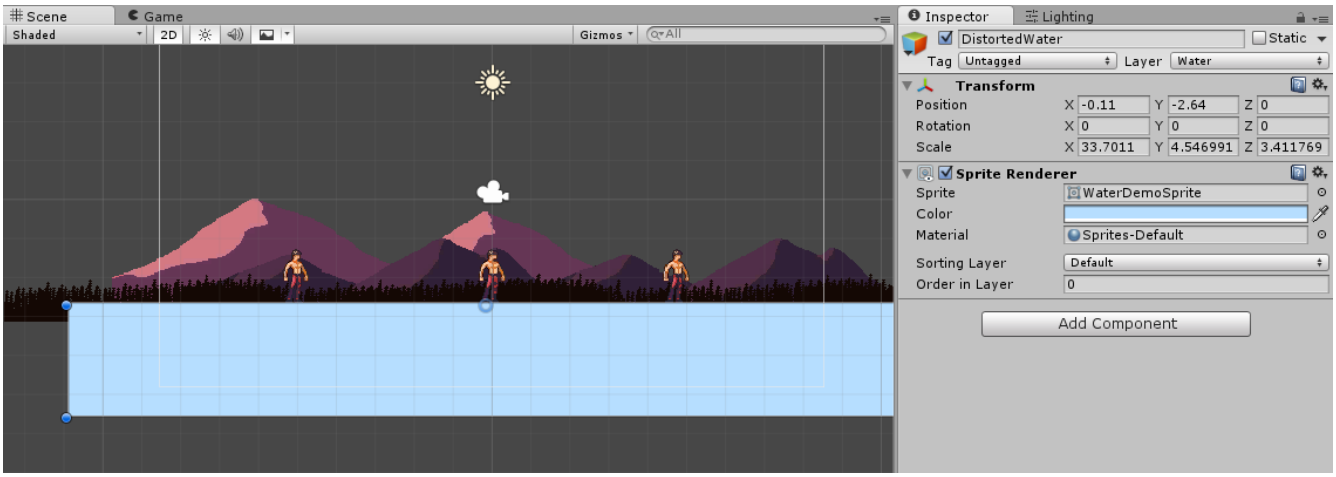


2- Render Based Reflections and Lightweight SRP support

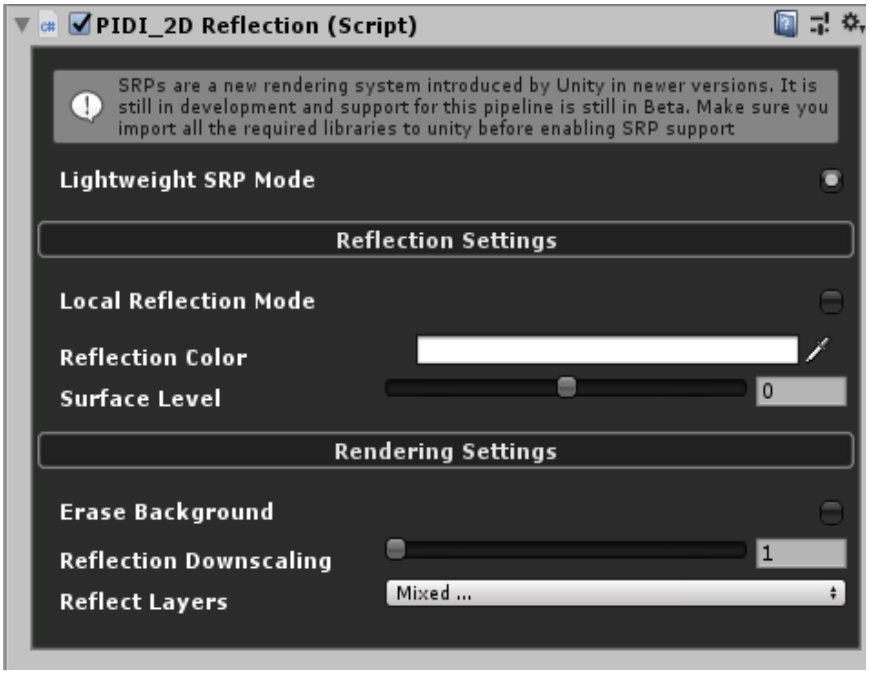
For more complex scenes where the sorting of different objects makes the previous workflow impossible or where a precise control of which objects get reflected and which are ignored is required we have implemented a secondary workflow through the PIDI_2DReflection component.

This component makes use of the shaders in the “Render Texture” branch and uses specific RenderTextures for each reflection, giving a great amount of control over the final results as well as avoiding most of the issues and conflicts caused by the scene's sorting and drawing order.

To create reflections with the PIDI_2DReflection component you must create a sprite with its anchor on the reflective edge of the surface just as in the previous method. Assign this sprite to a layer that can be ignored by the PIDI_2DReflection component (by default, the Water layer is ignored, so you can assign your reflection to this layer).



Once this is done, add the PIDI_2DReflection component to the sprite. The PIDI_2DReflection component has different settings and parameters to control depending on what you want to do. Here is a look at the basic interface for a basic 2D reflection :



This component has been completely revamped after version 1.2. For easiness of use, it now also includes the version number at the bottom so you can be sure which version of the tool you are using.

In Unity versions above 2018.1 the first setting available on the component is the Lightweight SRP Mode. Make sure to enable it immediately if you are targeting this rendering pipeline as failure to do so will spam your console with errors and mess with the rest of the rendering in the scene. **You must also add the PIDI_2DSRPCamera component to all the cameras in the scene that will see the reflections.**

The rest of the settings available on the basic interface are split on two sections : **Reflection Settings** and **Rendering Settings.** On reflection settings you can control the most basic options of how you want your reflection to look while the rendering settings control the internal working of the component and what exactly will be reflected.

Local Reflection Mode will treat this object as a local reflection and enable the Local Reflection section of settings on the interface.

Reflection Color changes the final tint of the reflections.

Surface Level changes the offset between the edge of the surface and where the reflection is being drawn from.

Erase Background removes the background around the reflections in selected shaders, mostly those dedicated to local reflections.

These values were available before on the material itself but since version 1.2 there were many internal improvements to the shaders and now works on a Renderer basis, allowing for many reflections to share the same material while having different colors, surface levels etc.

Reflection Downscaling allows you to reduce the quality of the final reflection to improve performance.

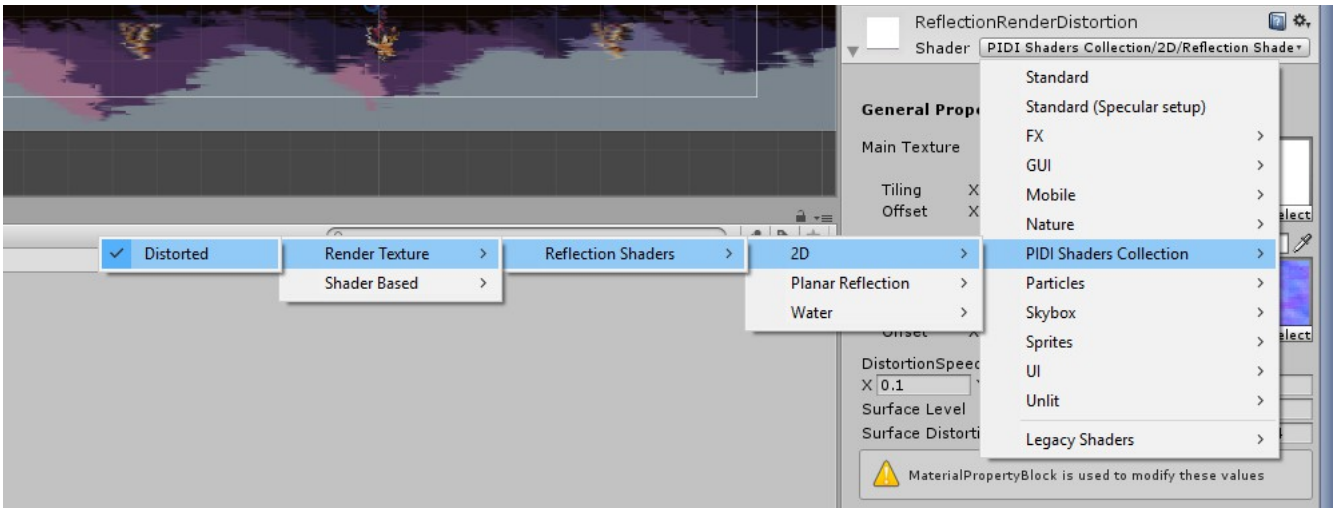
Reflect Layers is a layer mask that defines which layers will be reflected and which ones ignored.

Draw Over Layers. In mask-enabled shaders a variable for a second layer mask appears, letting you define over which layers will the reflection draw and over which it will be invisible.

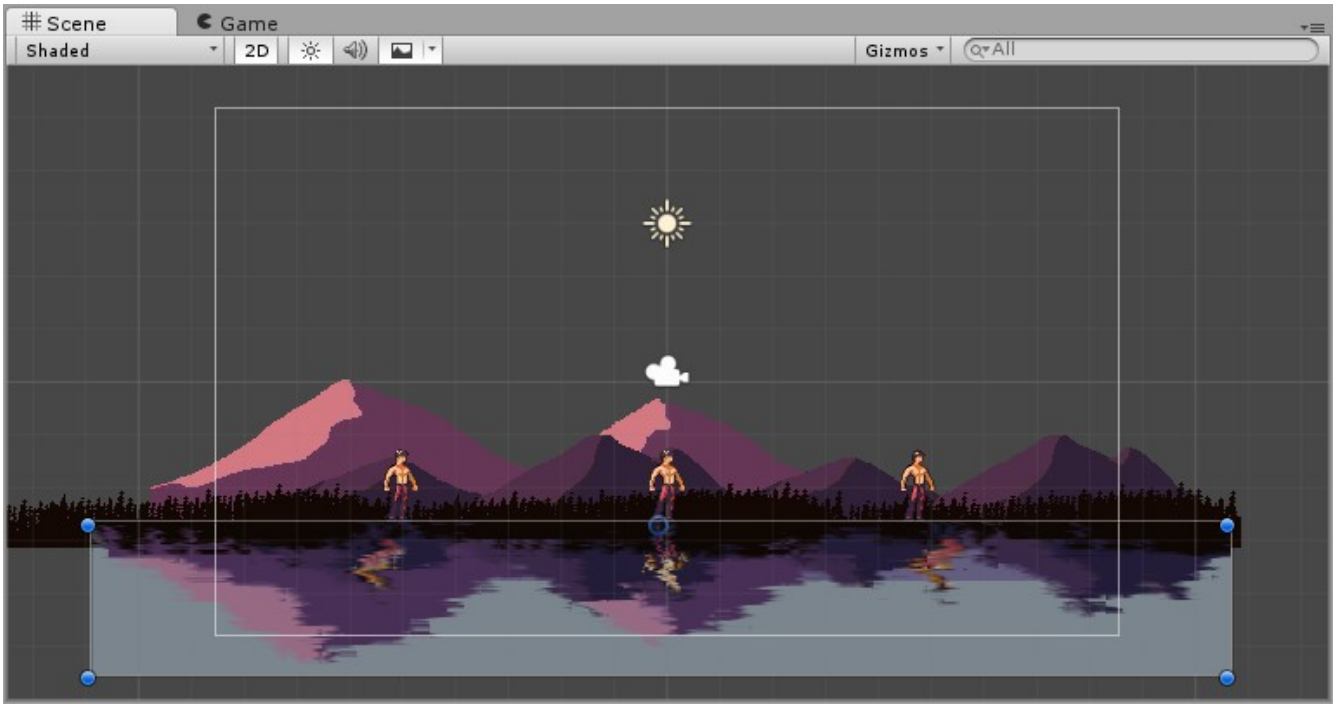
On top of this, the interface is full of tool-tips and help boxes to help you understand better how each part works.

Once the component has been assigned to your sprite, create a material with one of the “Render Texture” shaders included in our package. In this example, we will use one with support for distortion. We will also assign the waves map, the distortion strength and speed like in our previous example.

Important Notice : If you are working on the Lightweight SRP mode, ALL REFLECTIONS are rendered using the render method but you MUST use a Lightweight compatible shader. You will find them on “PIDI Shaders Collection/2D/Reflection Shaders/SRP/Lightweight/” when selecting a shader.



The reflection should begin displaying and rendering immediately.



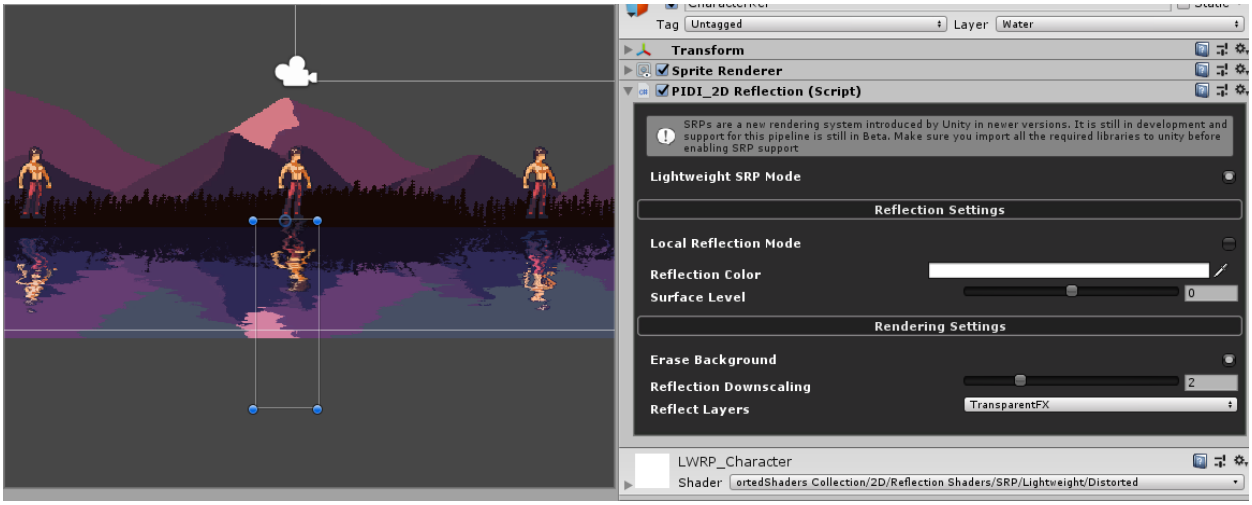
If you are targeting Lightweight SRP, due to the way it handles cameras and rendering of objects, you will see differences between the Scene and Game Views. While in Play Mode, only the Game View will have accurate reflections and when Play Mode is not enabled only the Scene View reflections will be rendered accurately.

This does not affect builds in any way, it is an Editor only behavior derived from differences between the new SRPs and the old rendering system that, at this moment, cannot be easily avoided.

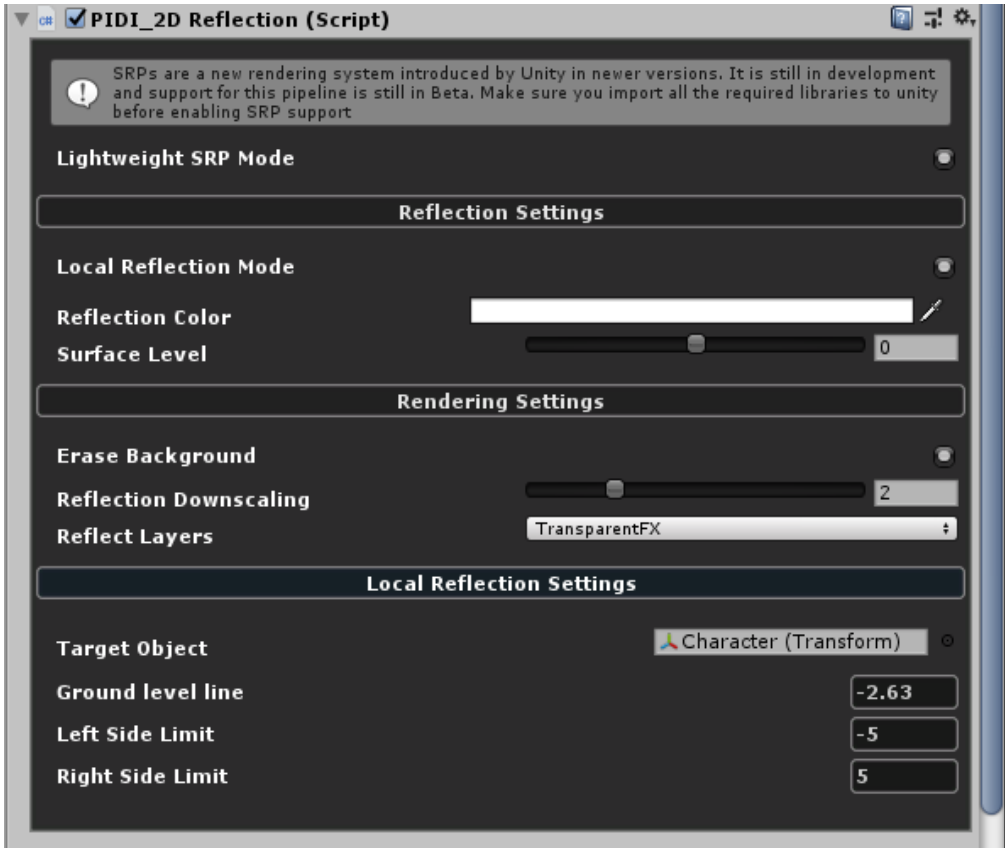
3. Local Reflections

Local reflections are reflection planes that follow and reflect a unique, specific object in the scene. When used carefully they are highly efficient, allow for very complex effects and compositions and give far greater control over the final results than other types of reflections.

Adding local reflections is very easy with this tool. The first step is to create a render based reflection as explained in the previous section and give it a small size that is slightly wider than the object it will reflect and twice as high.



Make sure that Erase Background is enabled and that the only reflected layers is the one where the object we will track is. After this, enable the Local Reflection Mode.



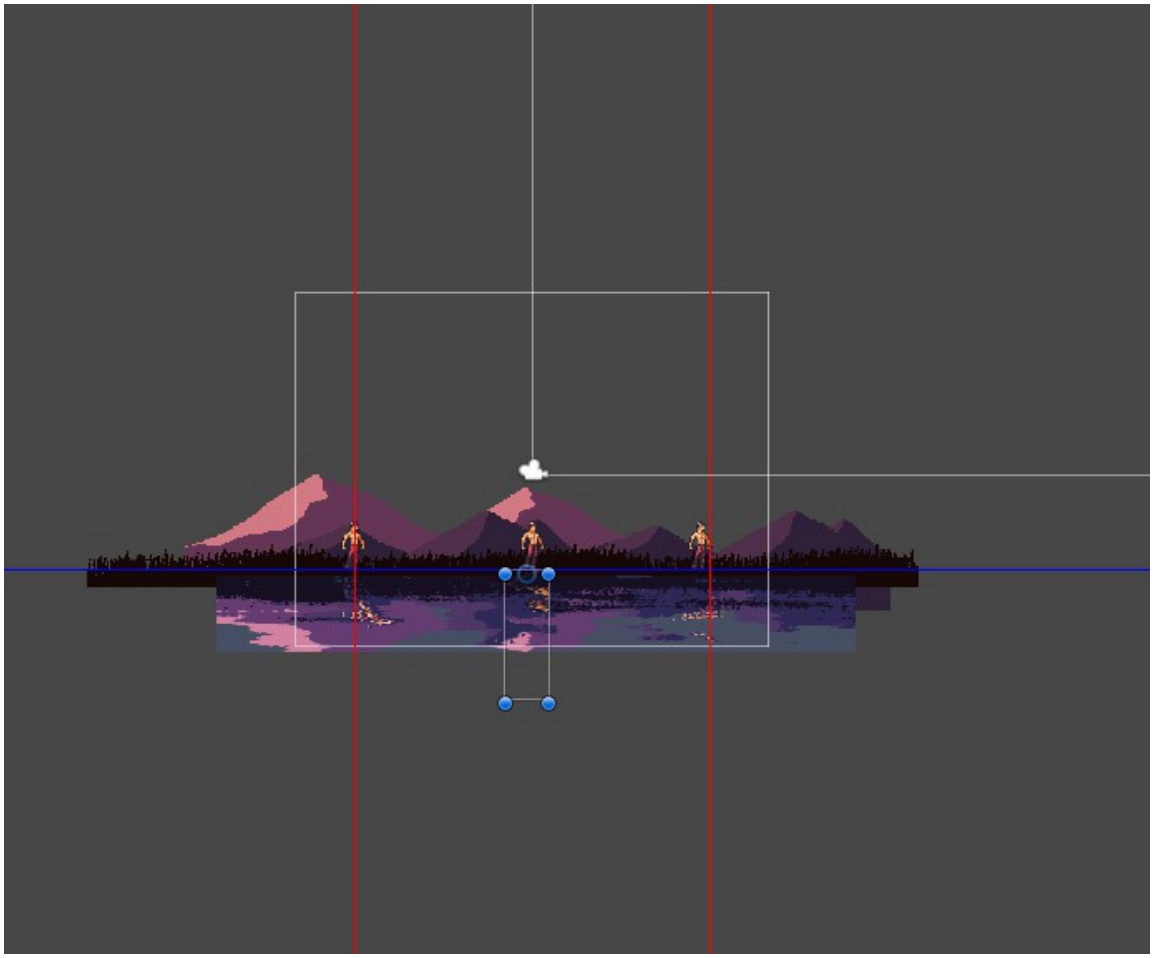
Enabling the local reflection mode will enable a new section of the interface. These settings allow you to have full control over the behavior of the horizontal local reflection.

Target Object is the object that will be followed by this reflection.

Ground Level is a line that defines the edge of the reflective surface. The local reflection cannot go above this line.

Left and Right side limits prevent the local reflection from going out of bounds after these points.

You can see the movement limits of the local reflection on the scene view :



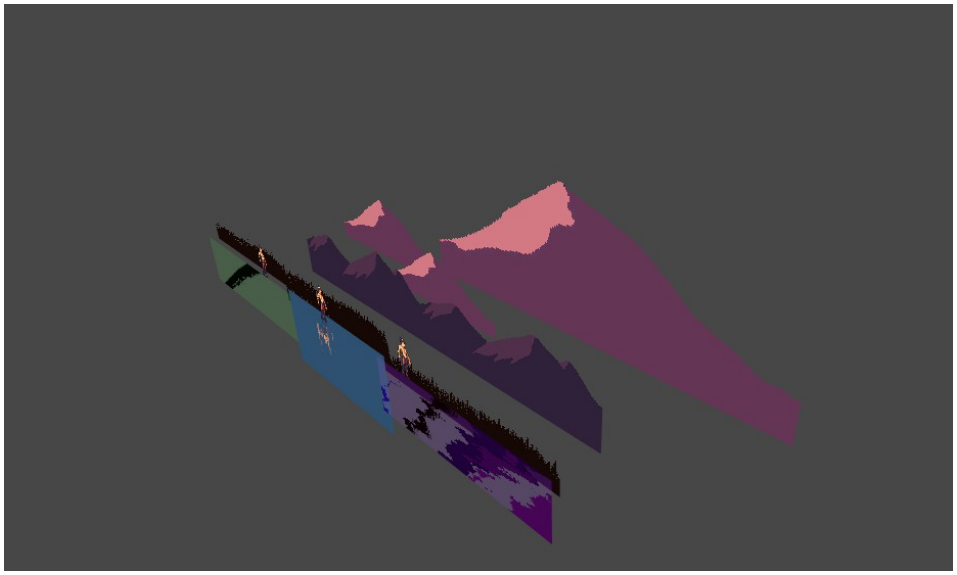
The ground level is displayed as a blue line while the left and right limits are displayed as red lines.

Once this is done, you just need to set your local reflection as a child object from the target object you want to reflect, place it in the precise position where you want it to be at runtime, and you are done.

Local reflections have been added to your character.

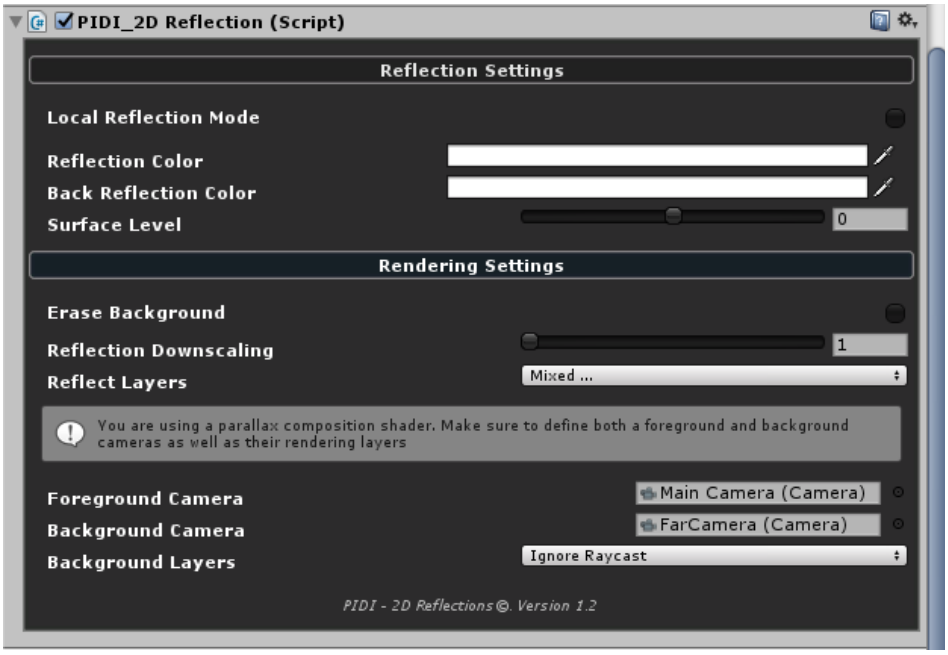
4. Parallax Reflections

1 - Deprecated Method



If your game is using parallax scrolling with different cameras being used for foreground and background you will most likely run into conflicts when setting up your reflections. To solve these issues a new shader and workflow has been added while keeping the simplicity and ease of use of the other shaders.

For your reflection material, instead of one of the standard shaders select the Render Texture/Distorted/Parallax Composite shader. Once this shader is assigned to the material, the PIDI 2D Reflections component will expose additional settings for you :



In this particular case, you will notice that a few new variables have appeared in the interface.

Back Reflection color refers to the color that will tint the reflection of the parallax background while reflection color will tint the reflections of the foreground.

Foreground Camera and Background Camera define the two cameras used in a standard parallax system. Each one can have their own perspective and orthographic projections and will be combined in real time through the shader.

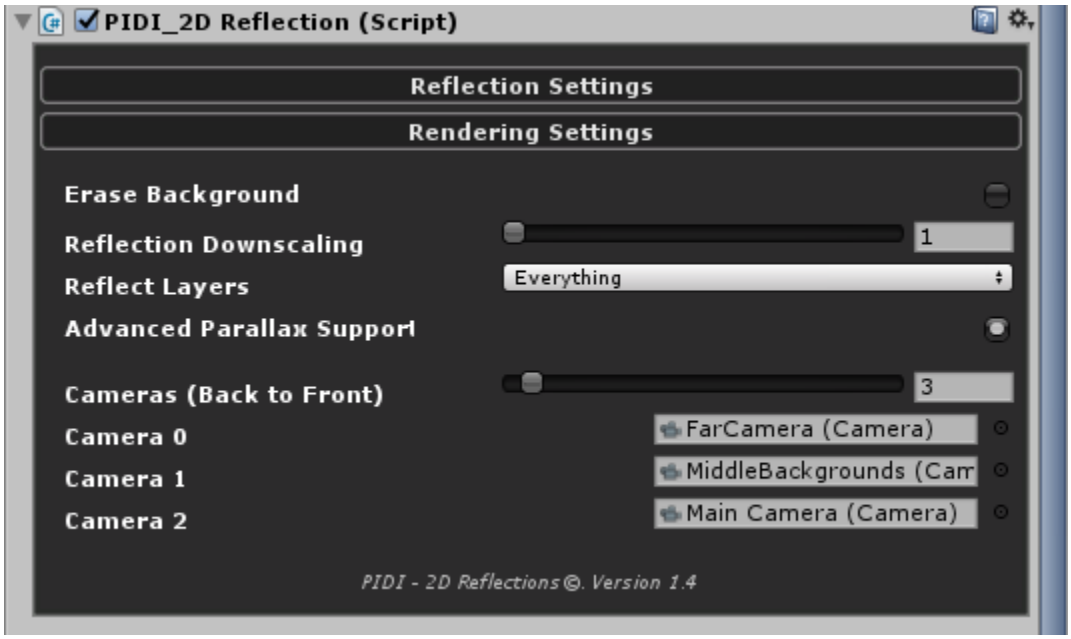
Reflect Layers will control which layers will the foreground camera reflect while background layers defines the layers that the background camera will reflect.

Once all these variables are set up, the component will work just as usual and all the other functionality will keep working as expected without any additional work on your part.

The new parallax composite shaders are also available for the Lightweight pipeline with a completely identical workflow.

1 - Improved Method

To use the new and improved Parallax support you just need to open the Rendering Settings tab of the PIDI 2D Reflections component and enable the “Advanced Parallax support” toggle on it. Then, assign your camera's to all the available slots that appear in the interface ordering them from back to front and assign to your sprite a material with one of the RenderTexture based shaders. No additional work is needed and the tool will automatically mix and combine the outputs of all your cameras to build an accurate, parallax compatible reflection. Up to 24 cameras are supported by this new method.



5. Final Notes

The tool is aimed at 2D game development and will not give acceptable results nor behave as expected in 3D games. The tool has been tested thoroughly both on DirectX and OpenGL environments, on desktop platforms (Windows/Linux) and Android mobile devices.

The performance of the tool depends entirely on the complexity of the scene and the amount of objects that will be reflected.

While this software has been thoroughly tested across all Unity versions from Unity 5.6to Unity 2018.1 and extensive optimizations and fixes have been included to avoid any bugs, the software is provided as is and without any explicit warranties. We strongly encourage you to backup all your work before making use of our tool or any other plugin from any provider.

If you have any troubles with our software or need any assistance, don't hesitate in contacting us at support@irreverent-software.com and we will get back at you with help.

If you need further tutorials and demonstrations, we recommend you to watch our two small tutorials on how to integrate PIDI 2D Reflections to your game in less than 2 minutes :

https://www.youtube.com/watch?v=jb3F78bY6g8

https://www.youtube.com/watch?v=2wR04jg4vtU

Thank you very much for purchasing our software, we hope our tools and the PIDI Game Development Framework will help you make awesome games!

The Irreverent Software Team.