Table of Contents

Features
Getting Started
Debug View List
Available Everywhere
Add Custom View
Troubleshooting
How-to
Changelog
Contact
Gallery

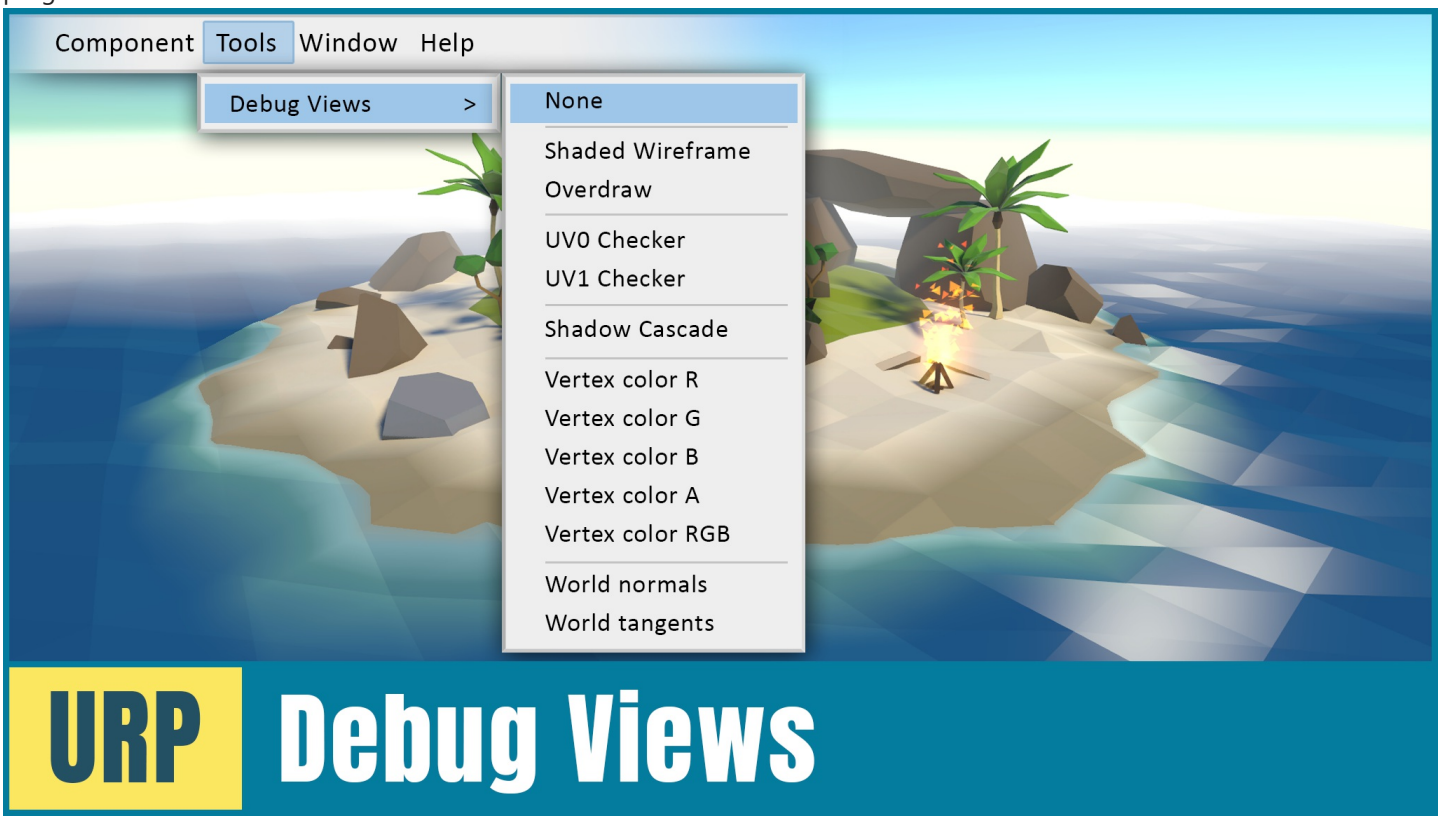
URP Debug Views features

URP Debug Views is a unique plugin that adds debug views support for LWRP / URP.

It adds most of the original views from the built-in renderer, news ones, and adding custom ones has never been easier!

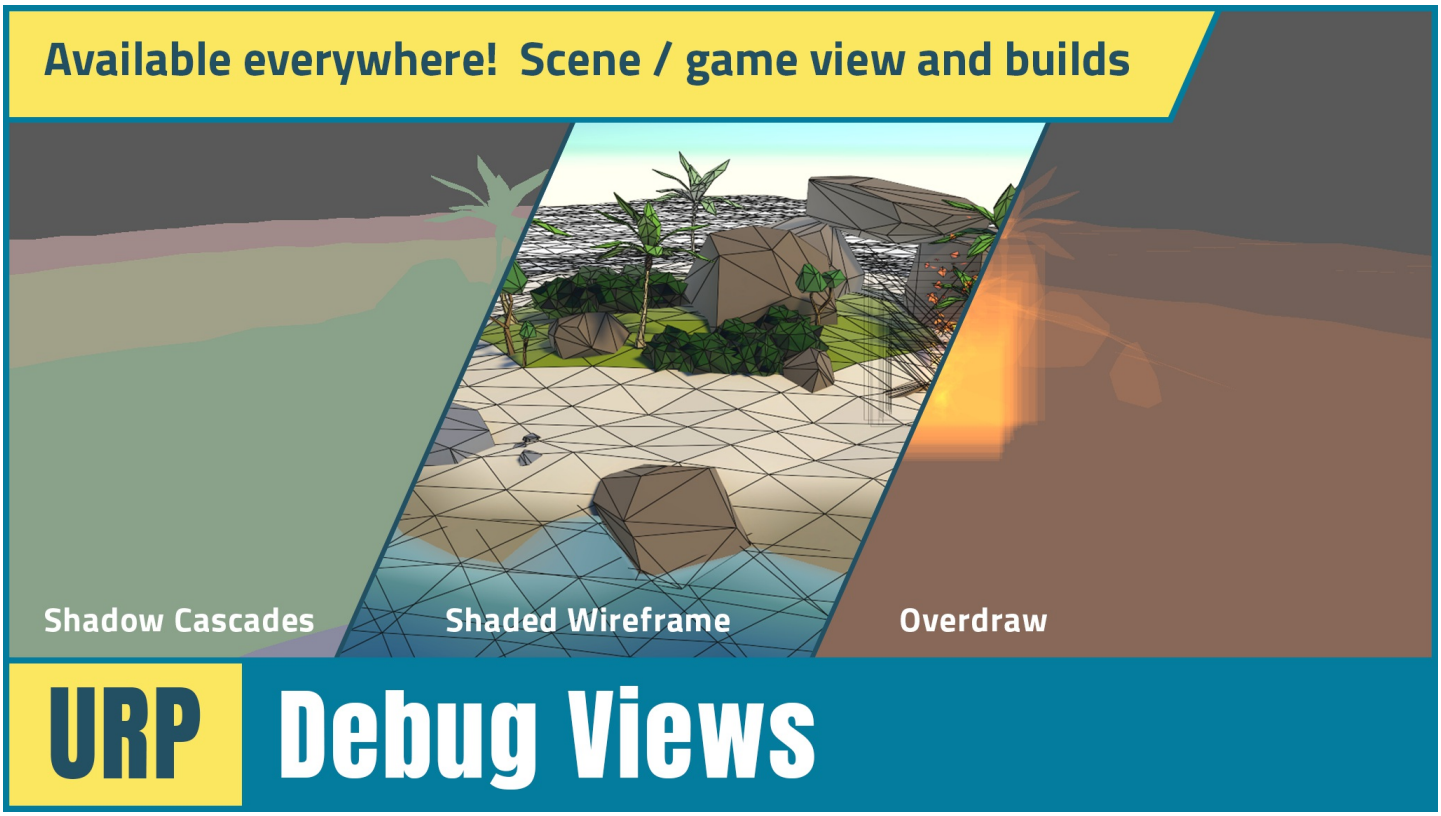
Available debug views

Several debug views from the built-in renderer are included, as well as several new one that where requested by artists and programmers!



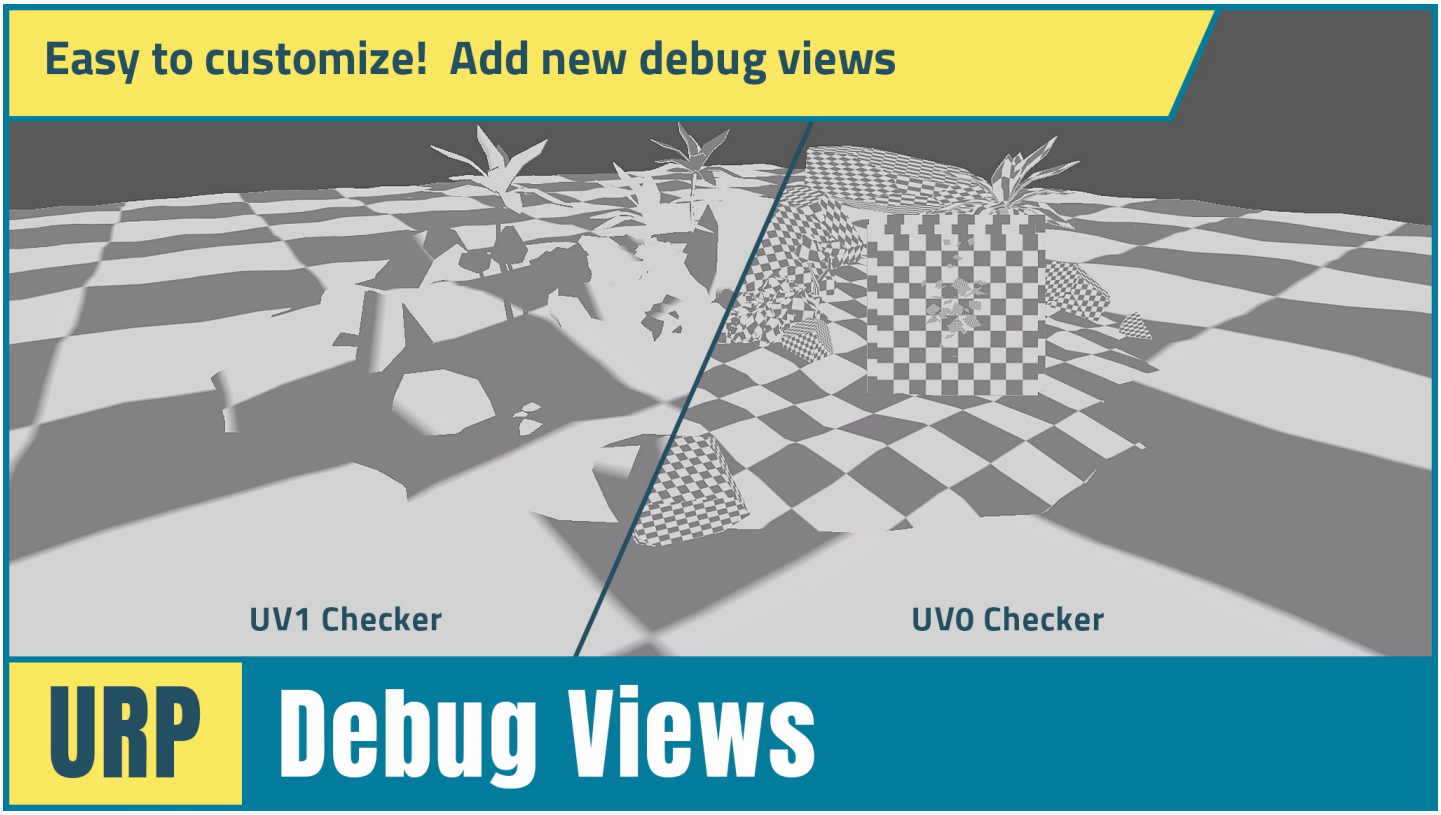
Available everywhere

Debug views work in scene mode, game mode and also in builds. It has been tested on various PCs, Mac, Android and VR.



Easy to customize

Adding custom debug views is very easy, a single shader is enough! You can then plug the newly created debug view in an editor menu, shortcuts or a custom in-game menu.



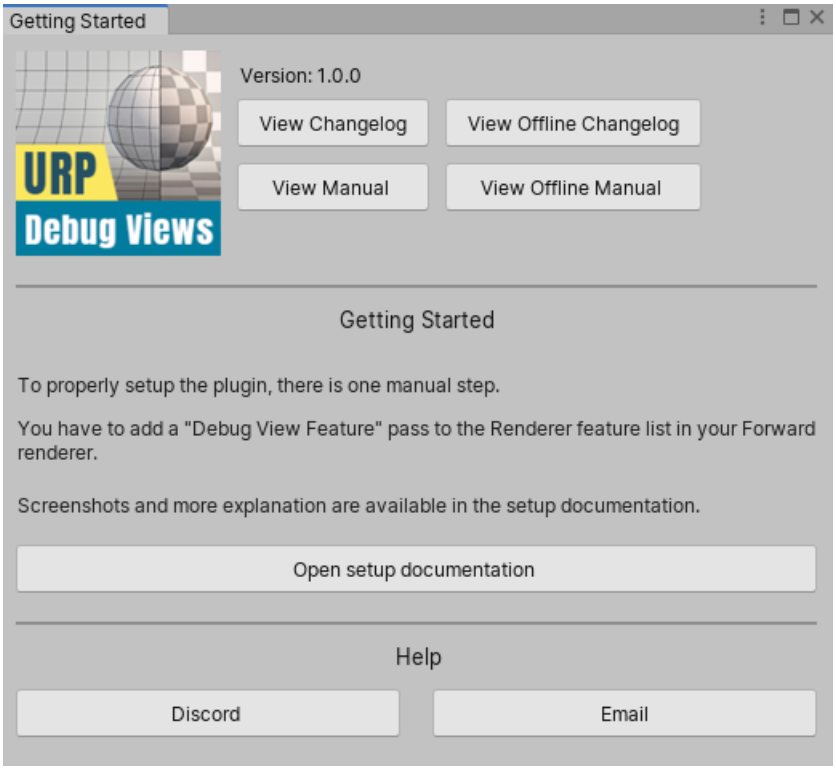
Getting Started

Warning

If you are updating from a previous version of URP Debug Views, please remove the URPDebugViews folder then re-download it. If you don't do this and I have removed some files, you could have some leftover files causing you troubles. I highly suggest that if you want to modify any script, or shader inside the plugin, you copy and rename it instead.

Plugin import

1. You are expected to have the LWRP / URP package in your project before you import the plugin.
2. At the moment, URP Debug Views **MUST BE** in the default location, right under the Assets/Plugins folder because of how the master node templates work. If you would like this to change, please contact me.
3. When importing the plugin, import everything.



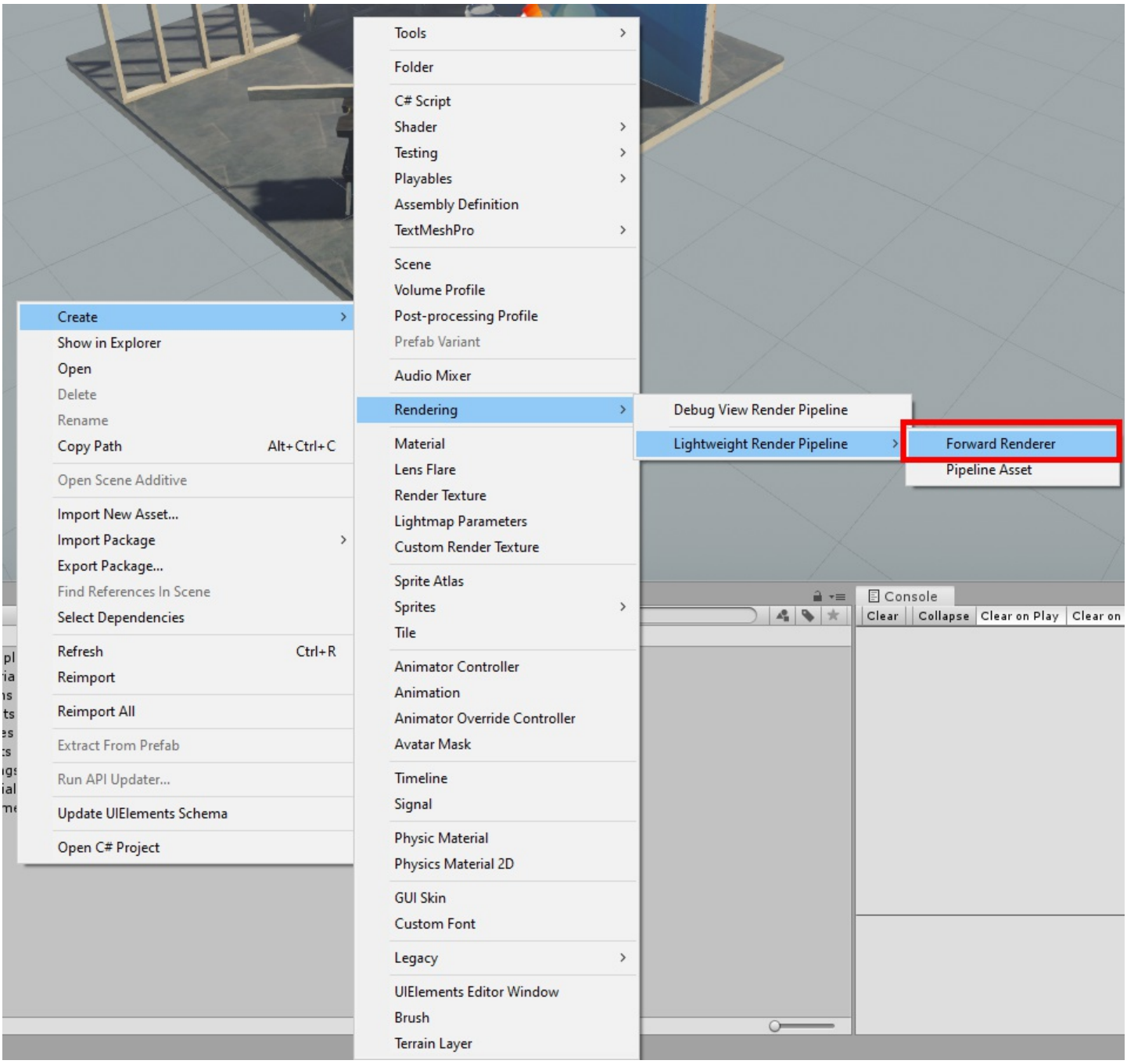
Setup

At the moment there is one manual step that you have to take to properly hook URP Debug Views in your renderer. Due to Unity's closed API, it's not possible to automatize it at the moment; but I am still looking in ways to improve this.

If you are on Unity 2019.1 or 2019.2

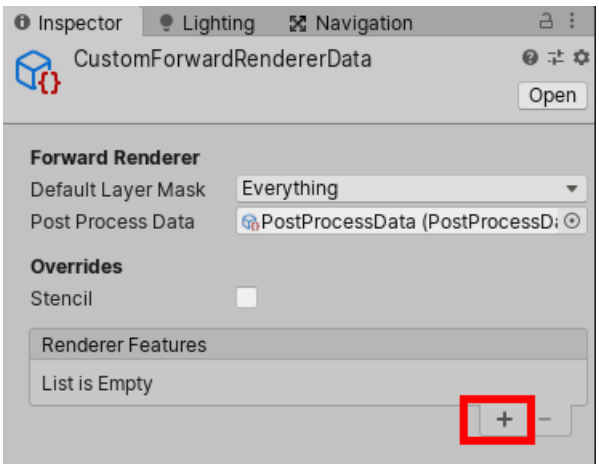
1) Create a new forward renderer object

Only if you don't already use one in your project.



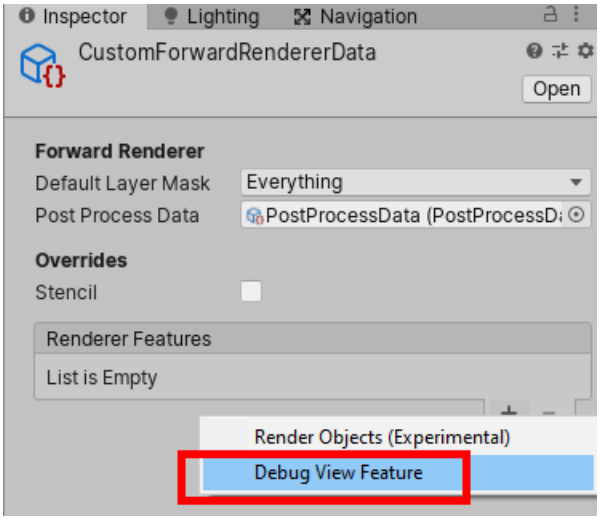
2) Add render feature to your renderer object

Then on this object newly created, add a new feature by clicking on the "+".



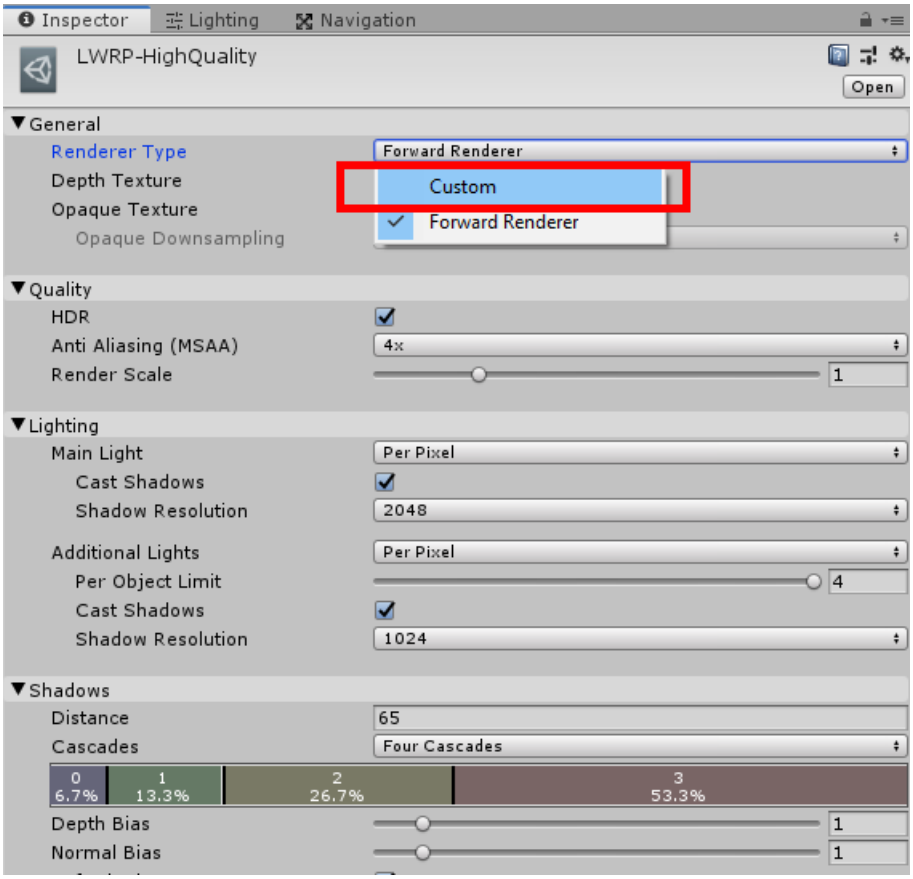
3) Select debug view feature

Select "Debug View Feature".



4) Add the forward renderer asset you created to your LWRP pipeline asset

Find your URP asset pipeline object in your project. If you can't find it, it is usually referenced in the project settings. In the "Renderer Type", select custom and reference the asset you created earlier.

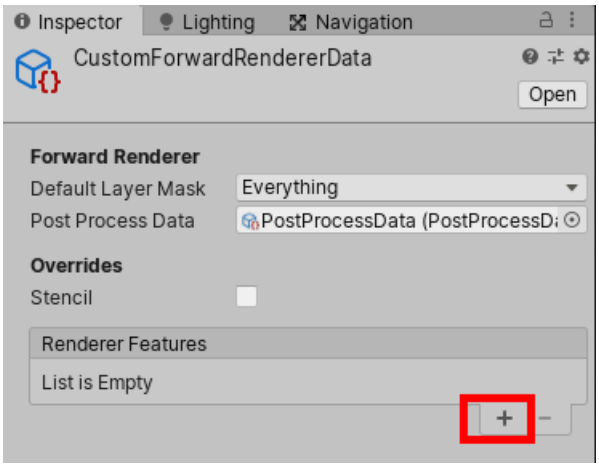


Done!

If you are on Unity 2019.3 or later

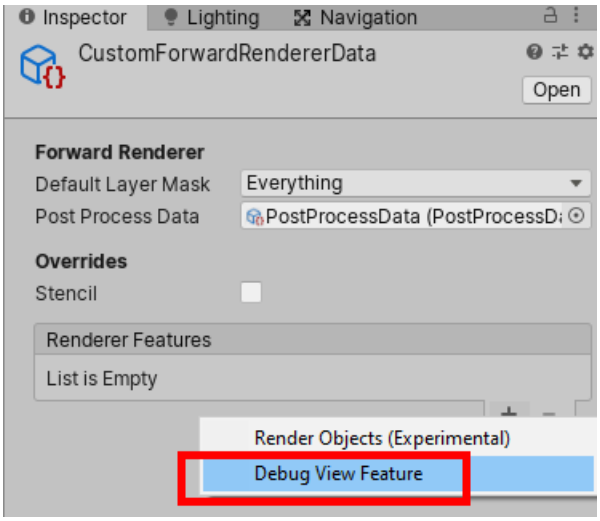
1) Add render feature to your renderer object

Find your URP asset pipeline object in your project. If you can't find it, it is usually referenced in the project settings. Then the asset pipeline object itself will have a "Forward Renderer" data linked. Then on this object , add a new feature by clicking on the "+".



2) Select debug view feature

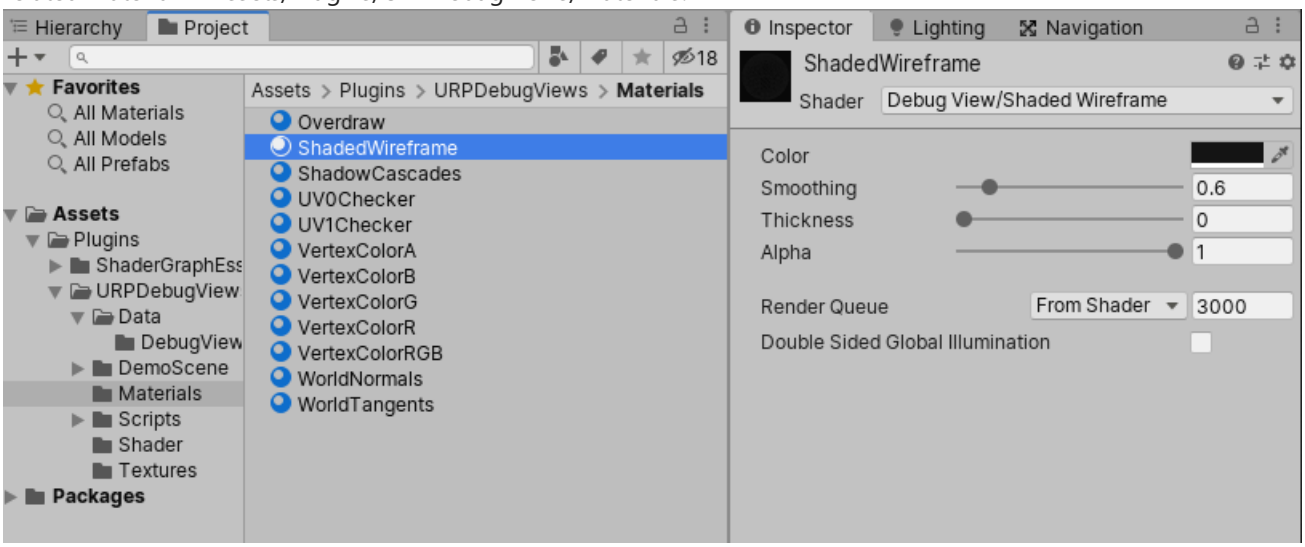
Select "Debug View Feature".



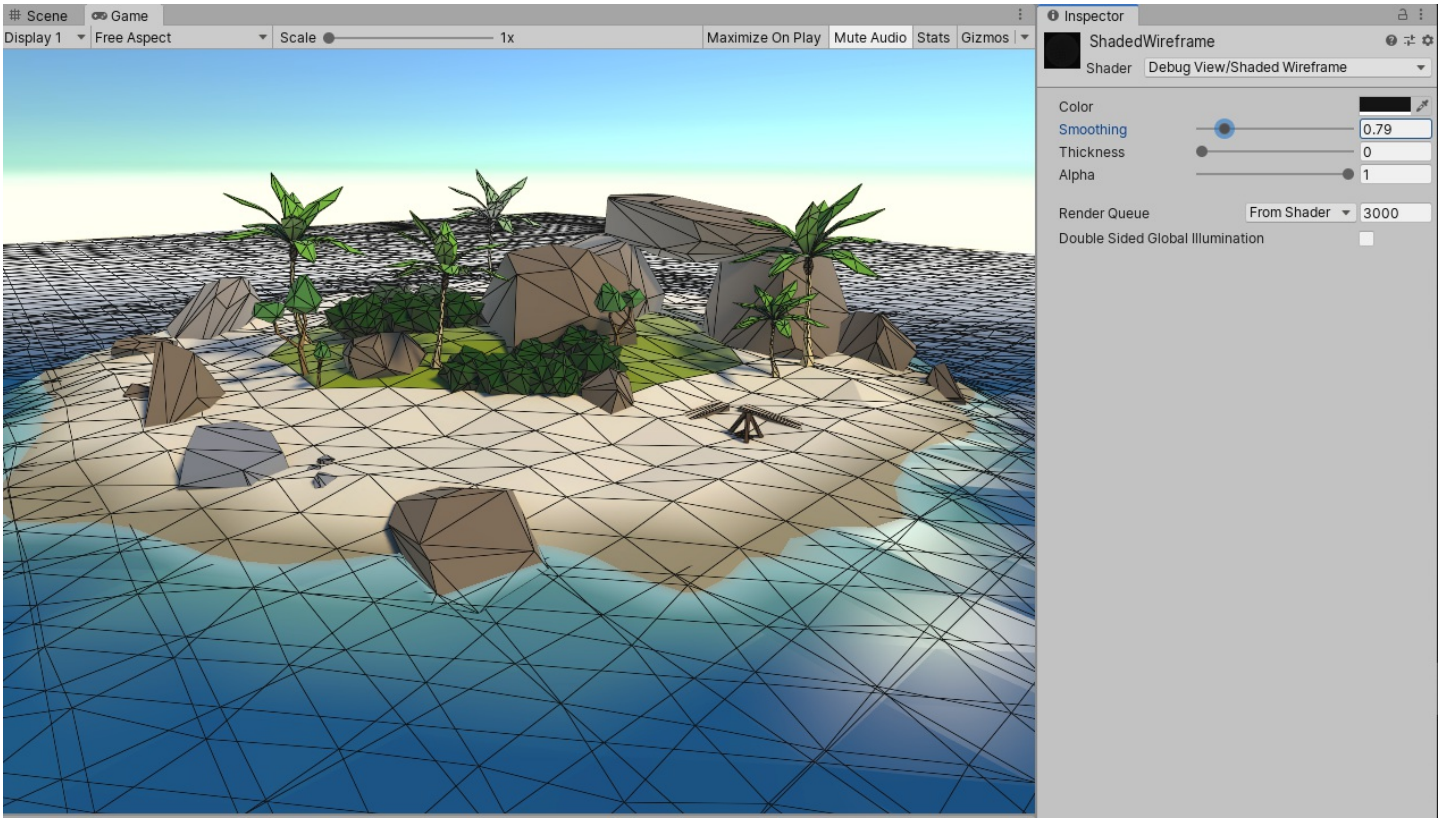
Done!

Debug View List

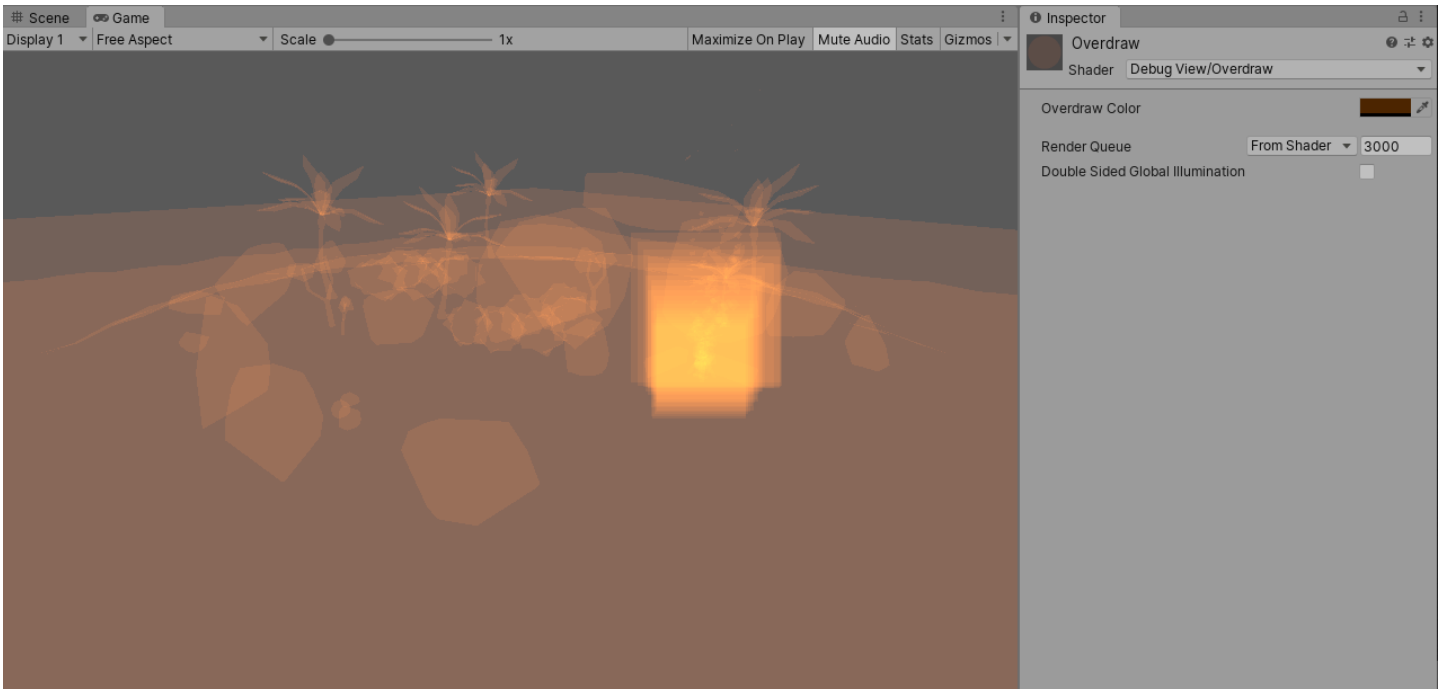
Here is a list of all available debug views in the plugin. Some debug views have parameters that can be changed by updating their related material in Assets/Plugins/URPDebugViews/Materials.



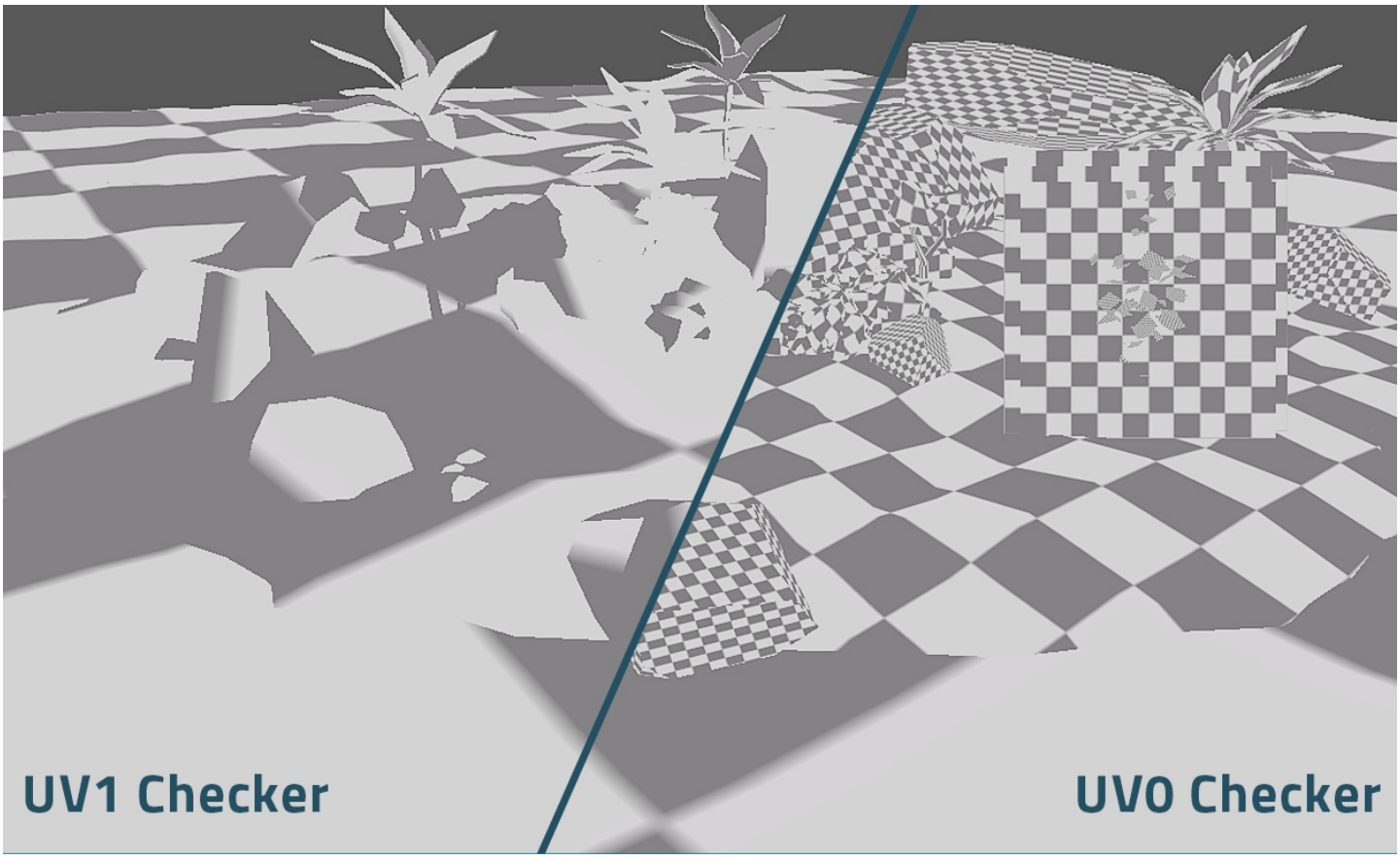
Shaded wireframe



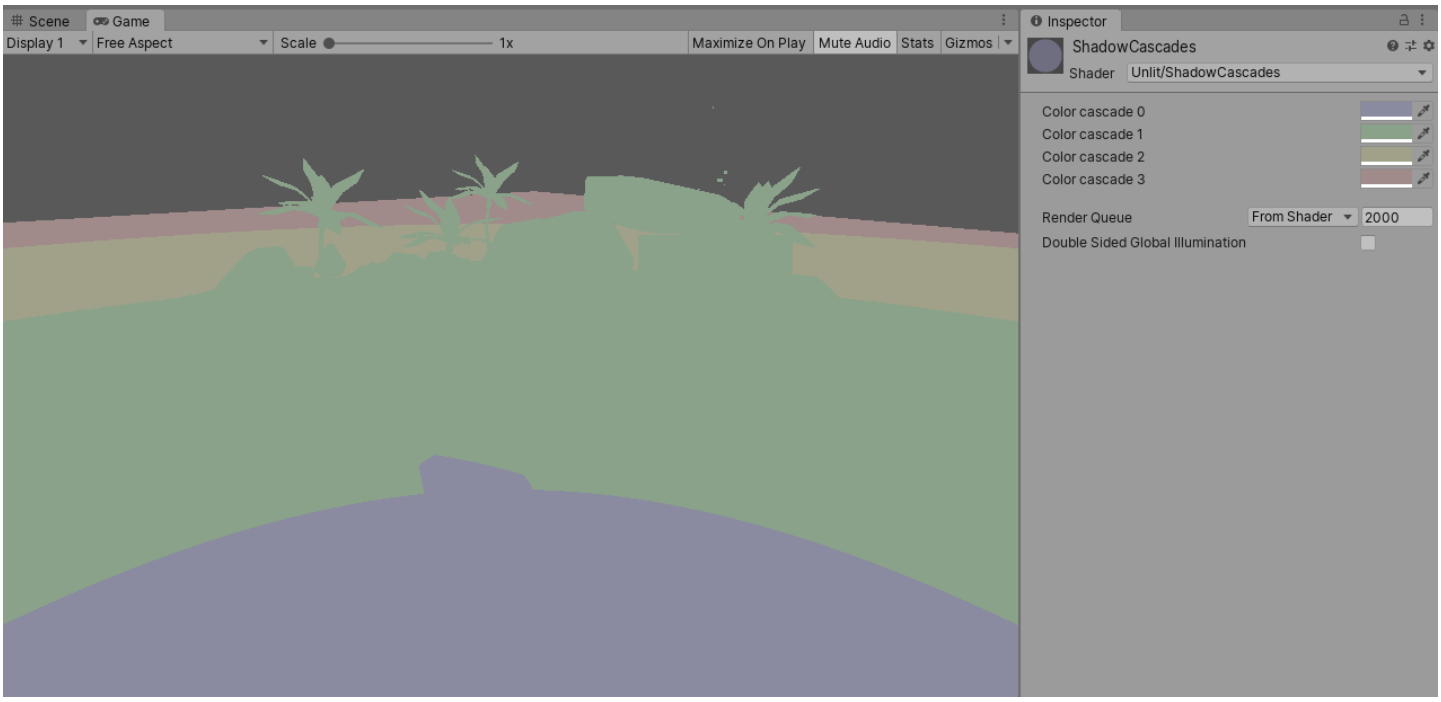
Overdraw



UV0 / UV1 Checker

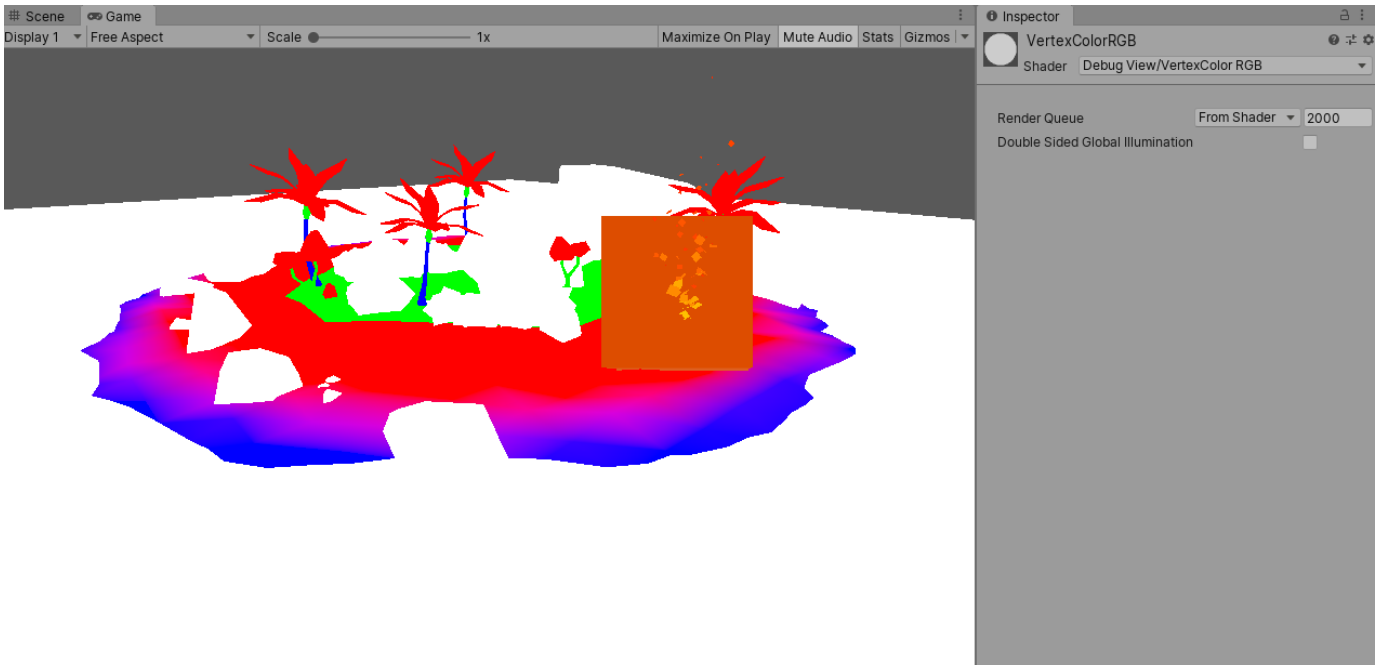


Shadow Cascades



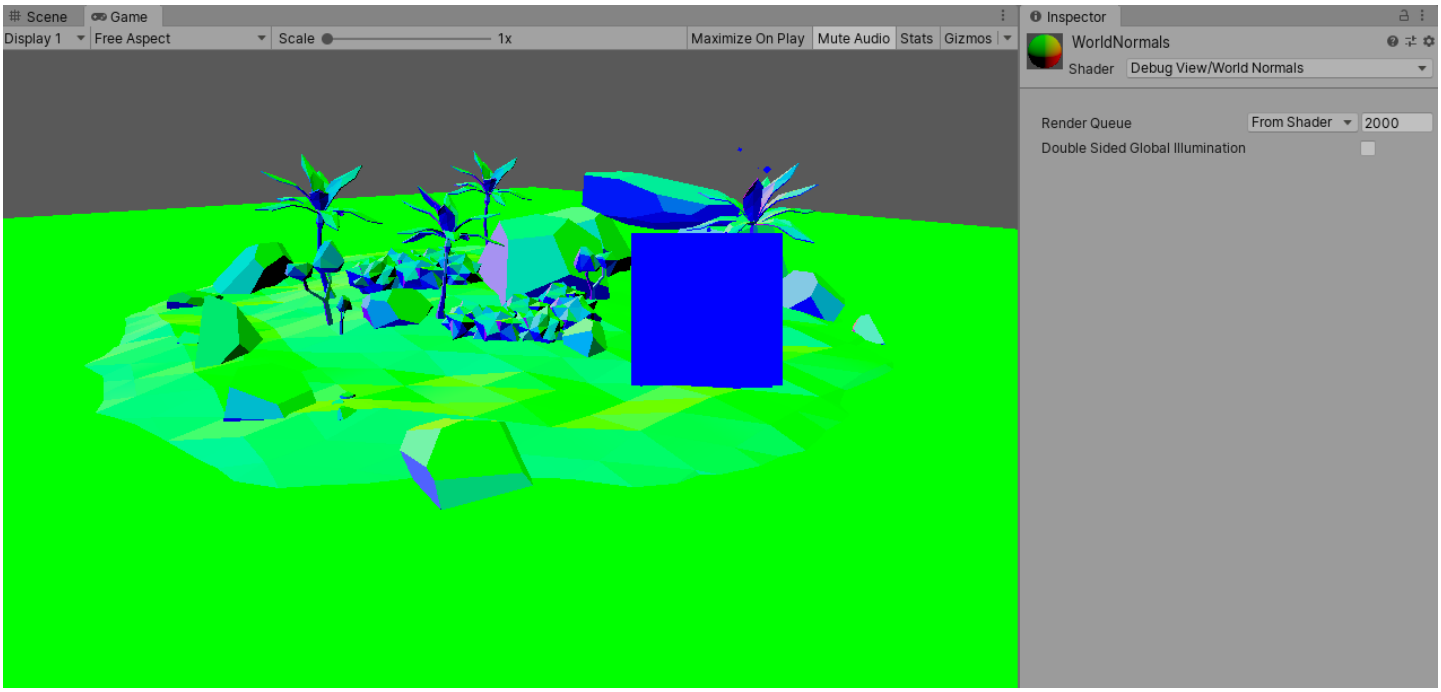
Vertex color

- Single R channel
- Single G channel
- Single B channel
- Single A channel
- Combined RGB channels

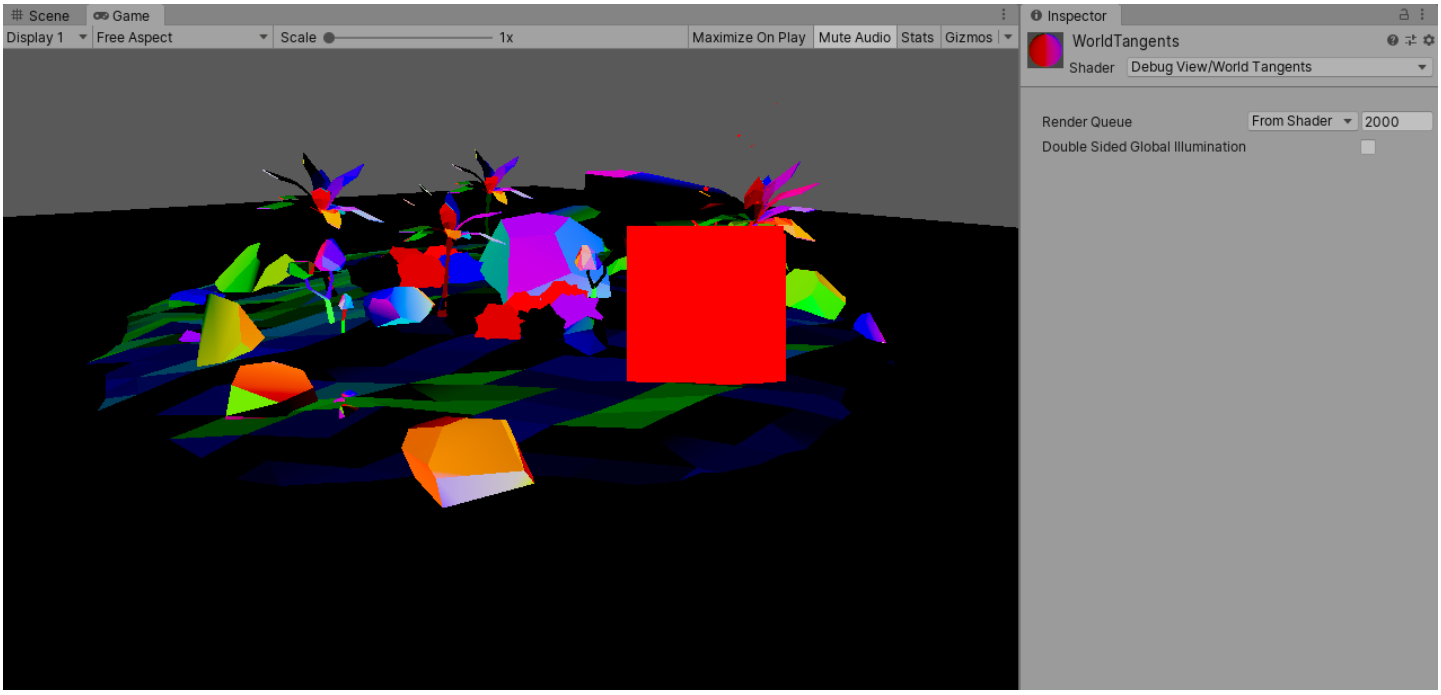


World normals

Mesh normals (doesn't include normal maps).



World tangents

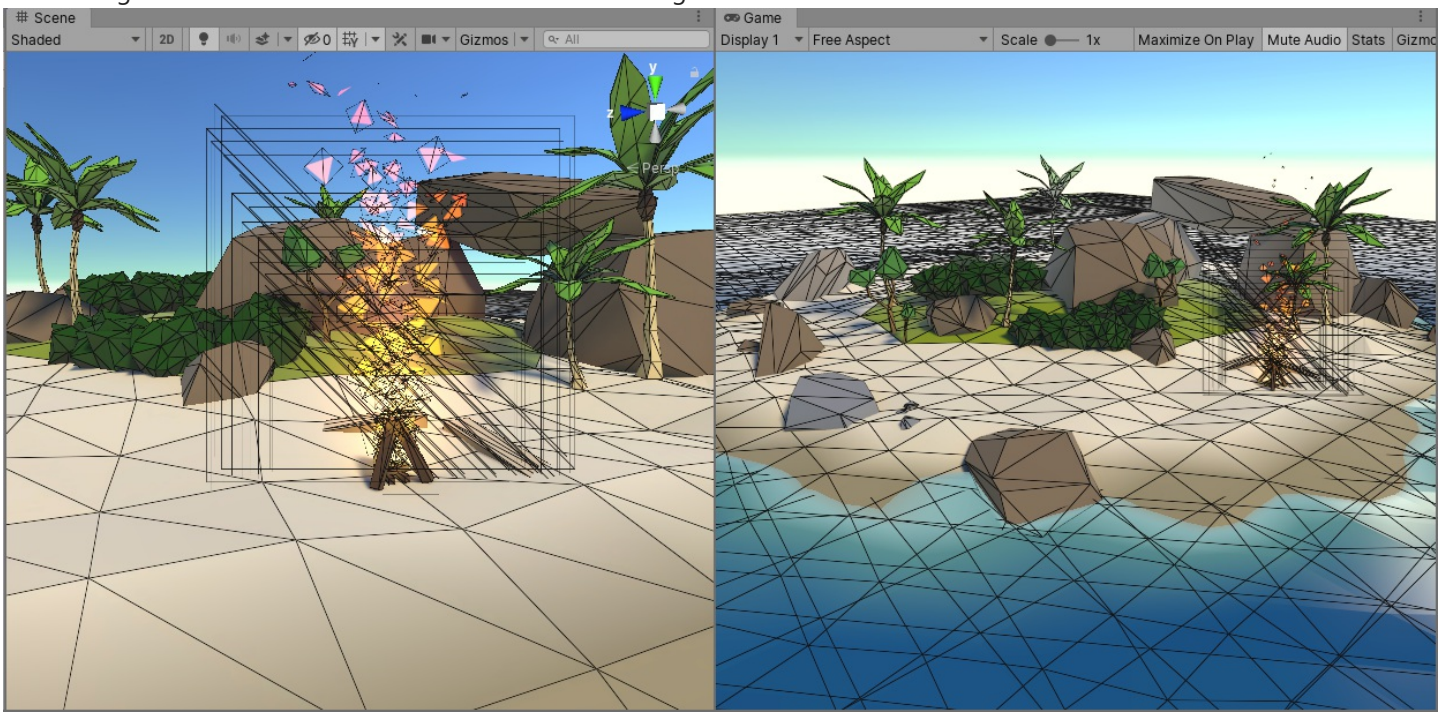


Available everywhere

Debug views are available in scene view, game view and builds (tested on various PCs, Mac, Android and VR).

Scene view and game view

URP Debug views work out of the box in both the scene and game view.

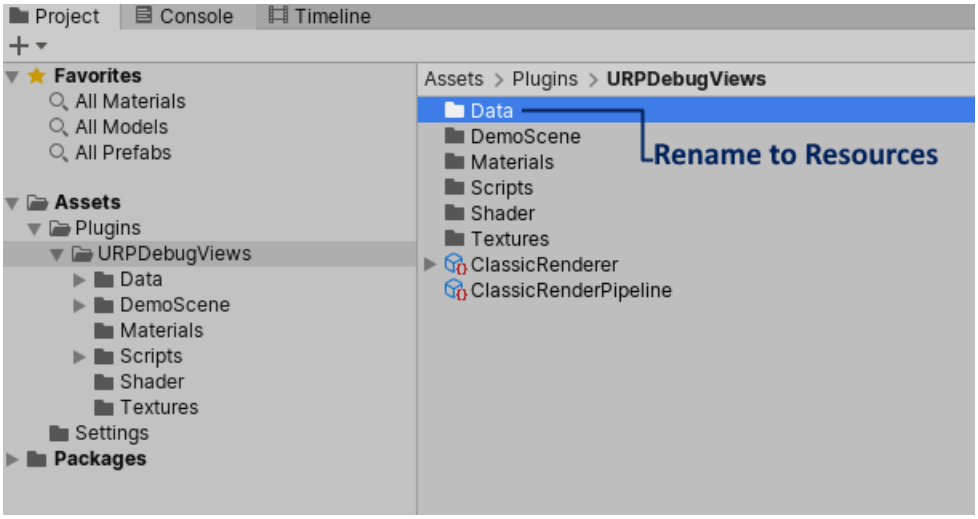


Builds

URP Debug Views isn't accessible in builds out of the box. That is because it would have to always include material and shaders for all debug views. This has a certain weight (especially for mobile platforms), which is why it is optional and off by default. Don't worry, it's super easy!

Rename the "Data" folder to "Resources"

Rename the "Data" folder to "Resources" so the plugin can access all data, material and shaders in builds.



Turn on debug views by code

You can access the API and link debug views to keys, touch control or your own debug menu. A sample of the code would be:

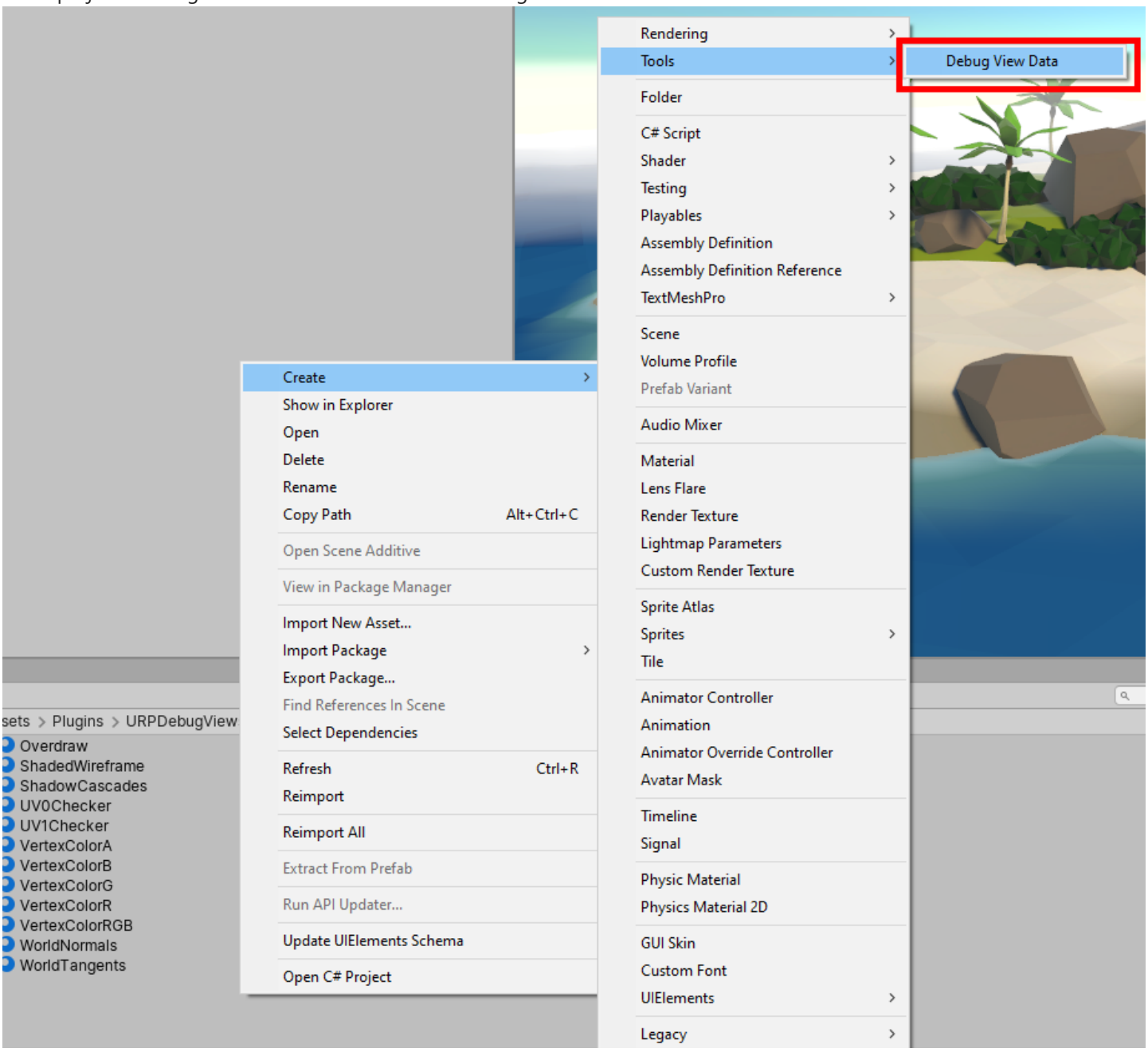
```
public class ChangeDebugView : MonoBehaviour
{
    private int _currentDebugViewIndex = 0;

    void Update()
    {
        // right arrow key will go through all debug views
        if (Input.GetKeyDown(KeyCode.RightArrow))
        {
            // select next index
            int nextIndex = _currentDebugViewIndex == DebugViewsManager.Instance.AllAvailableDebugViews.Count
- 1
                ? 0
                : _currentDebugViewIndex + 1;
            _currentDebugViewIndex = nextIndex;
            // use this call to enable a specific view at an index
            DebugViewsManager.Instance.EnableView(_currentDebugViewIndex);

            // if you prefer to hardcode the string
            // you ou can also access using the material name
            // DebugViewsManager.Instance.EnableViewWithMaterialName("ShadedWireframe");
        }

        // space key will enable / disable debug views
        if (Input.GetKeyDown(KeyCode.Space))
        {
            if (DebugViewsManager.Instance.CurrentViewData)
            {
                // pass null if you want to disable debug view rendering
                // and revert back to your normal rendering
                DebugViewsManager.Instance.EnableView(null);
            }
            else
            {
                DebugViewsManager.Instance.EnableView(_currentDebugViewIndex);
            }
        }
    }
}
```

Add a custom debug view

Adding a custom debug view is very easy! Follow the steps:

1) Create a new debug view data

In the project view Right clic and "Create -> Tools -> Debug View Data".



2) Fill the data

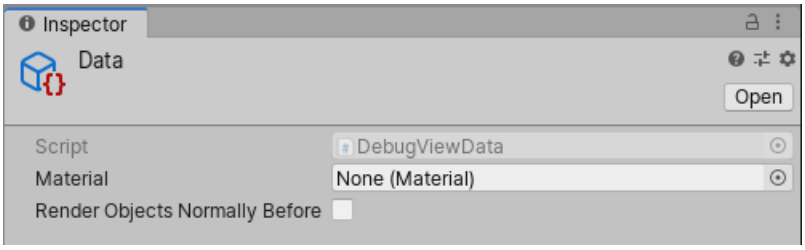
Now fill the data object you just created. It only needs

- A material that will be applied to all objects when rendering.
    - Please note that debug views replace existing materials with the provided one. As such, you can use any texture or original material parameter from the object rendered.

- *Render Object Normally Before*: if off, all objects in the scene are going to be drawn with the provided material, that's it. If it's on, all objects in the scene are going to be drawn with their own material first, then drawn again with the provided material.
    - If your material is opaque, then you have no reason to tick this checkbox as you will drawn on top of everything else

anyway.

- As a side note, this is how the shaded wireframe is done.



> 💡 Tip
>
> The material can use a hand-written shader or a shader from ShaderGraph, it doesn't matter!

Troubleshooting

If all debug views are working except for Shaded Wireframe

It most likely that the setup has been properly done. Please have a look at the setup instructions

How-to

How to add a custom debug views

Have a look at the guide here.

How to have the debug views working in builds

Have a look at the guide here.

Changelog

1.0.3

Submitted on July 25th 2020.

- Added support for Unity 2020.1.

1.0.2

Submitted on February 9th 2020.

- Fixed a bug that could get the editor stuck in debug view mode when changing a script.

1.0.1

Released on January 13th 2020.

- Fixed a bug that prevented shaders from ShaderGraph to be used in a debug view.
- Added asmdefs.

1.0.0

Released on January 9th 2020.

- Initial release

Contact

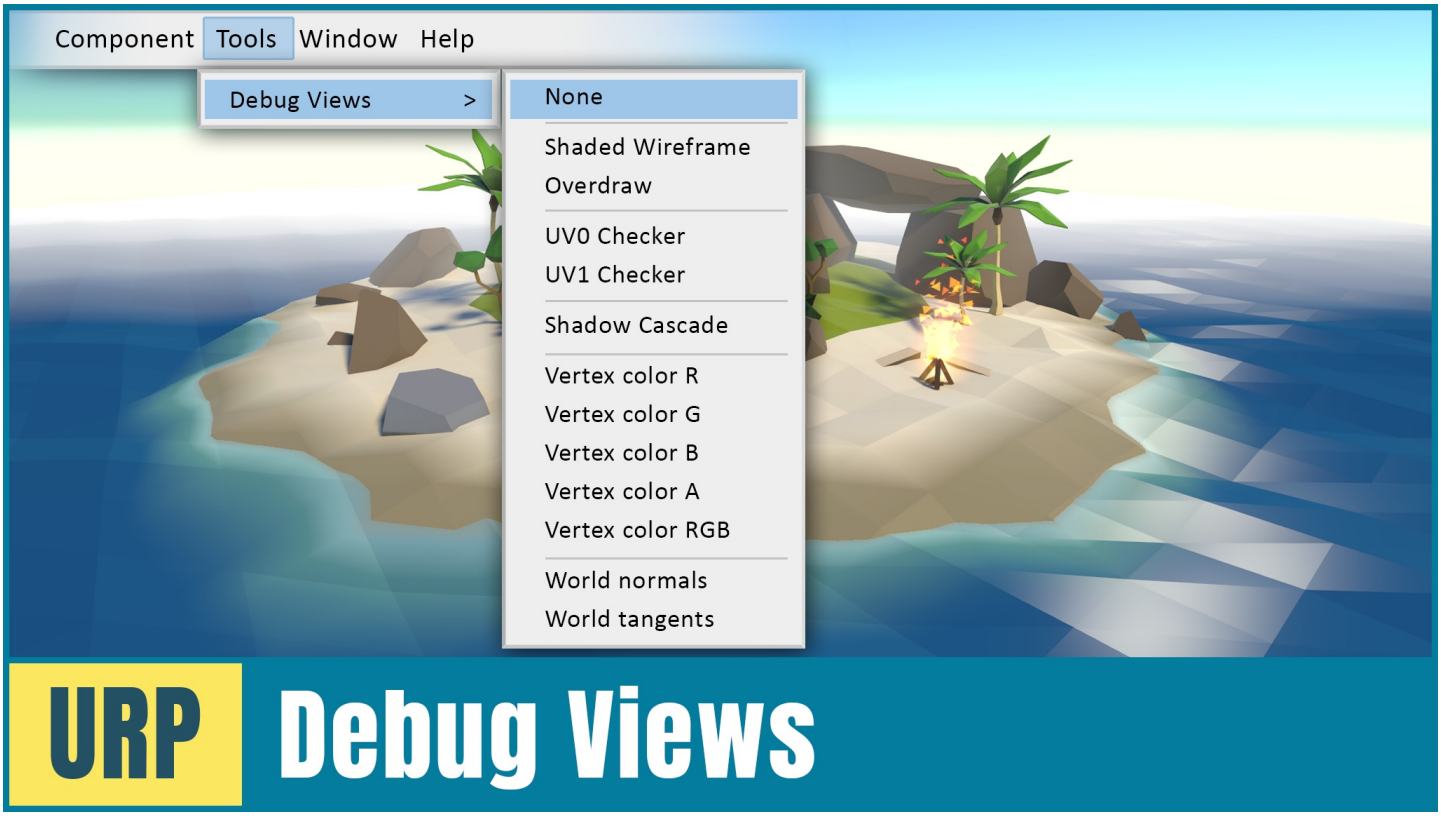
Don't hesitate to contact me about bugs, feature requests, or any question you might have before buying ShaderGraph Essentials.

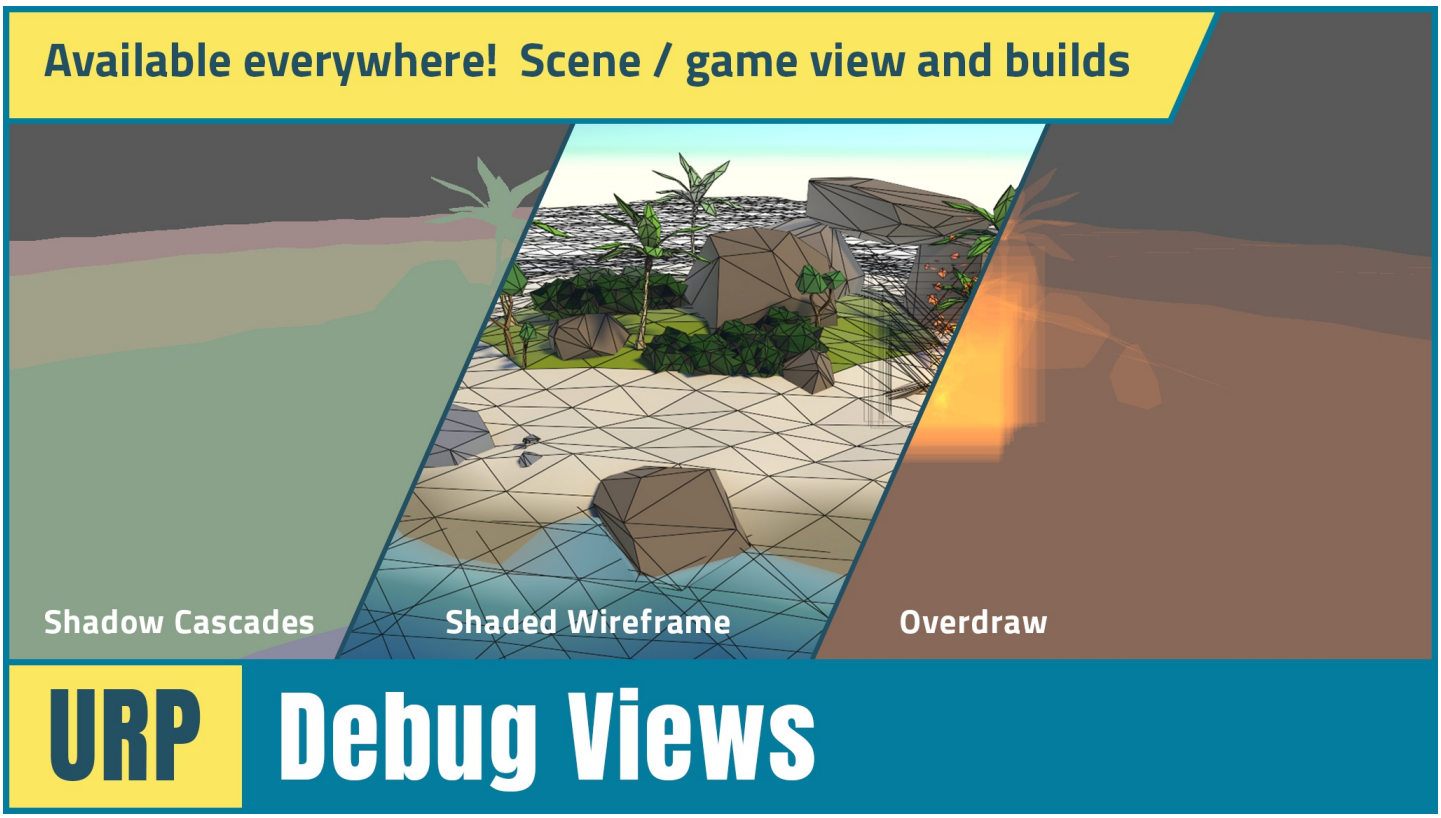
The fastest way to contact me is through the ShaderGraph Essentials discord server. You can also send me a private message from there.

You can also contact me by email:

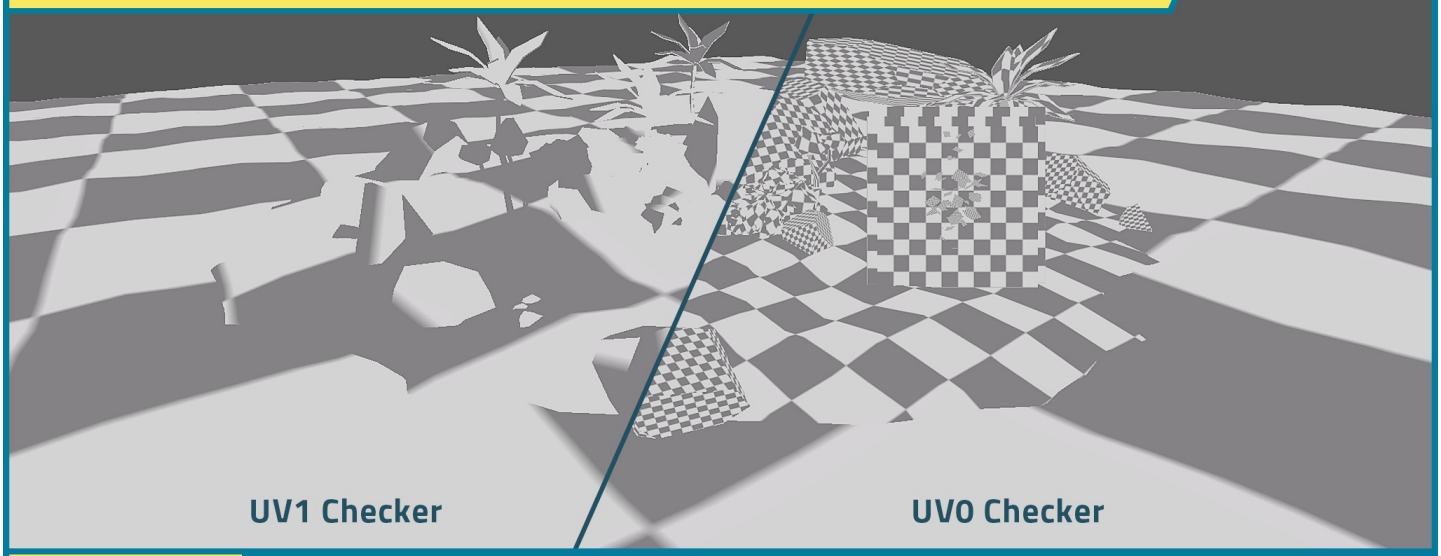
- about ShaderGraph Essentials at ph.graphics.unity@gmail.com
- for other business inquiries at info@phbarralis.com

URP Debug Views gallery

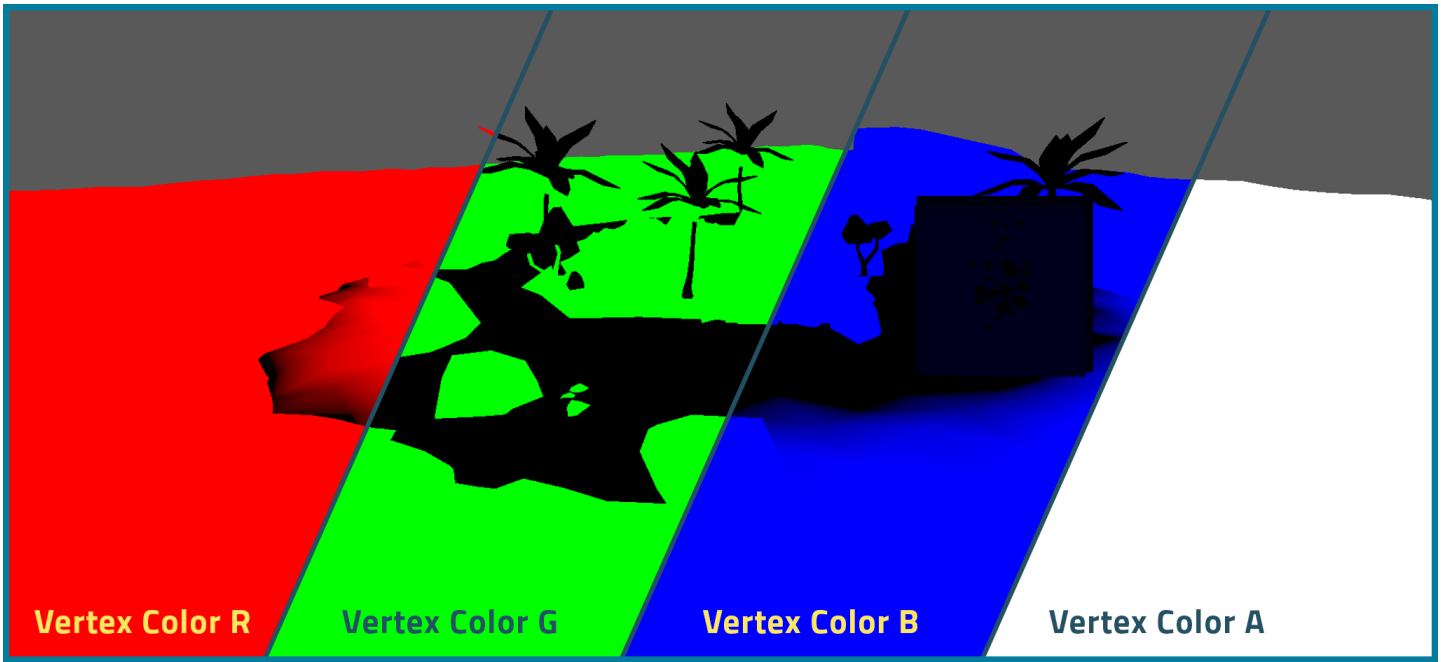




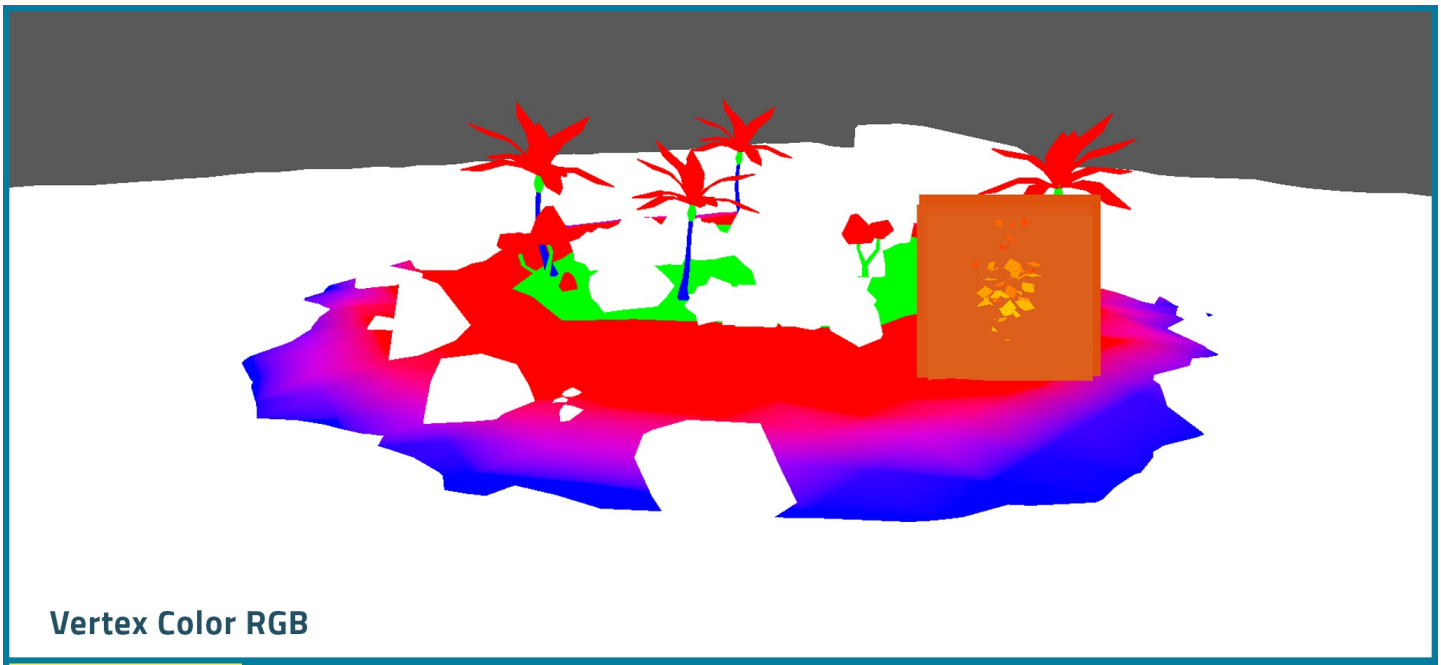
Easy to customize! Add new debug views



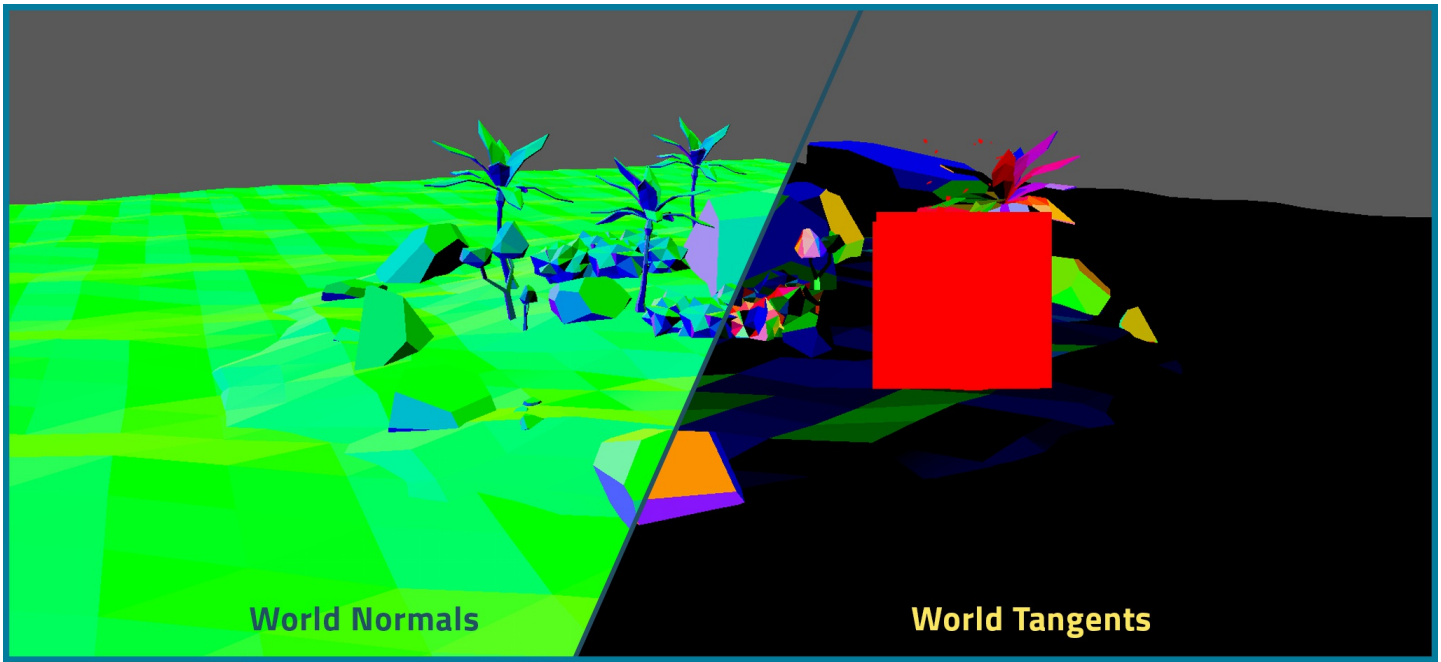
URP Debug Views



URP Debug Views



URP Debug Views



URP Debug Views