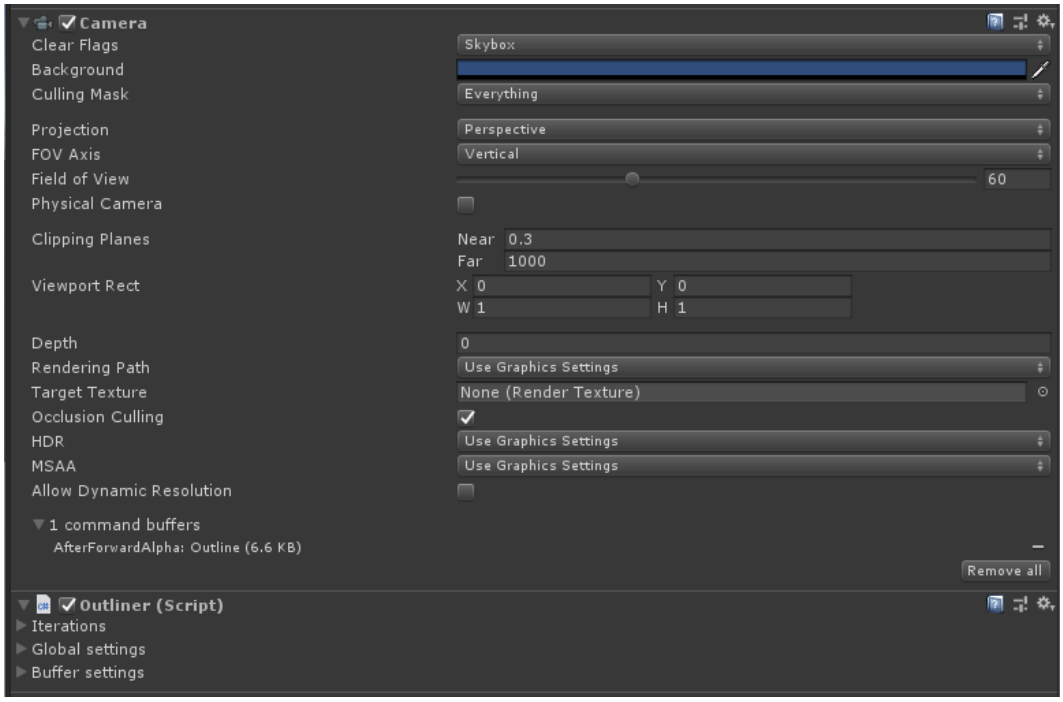
New information is marked in blue

Easy Performant Outline

Easy performant outline – is asset that provides high quality 3D outlining for Unity.
It supports various outline modes and features.

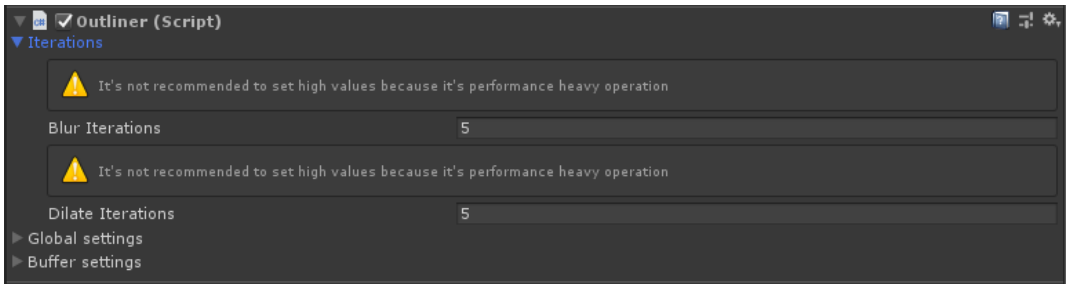
Components

Outliner: Component that should be attached to camera in your scene (if you are using URP, you can skip this explanation as it has a different setup than default renderer. At the end of the manual you'll see how to setup URP)



Let's look at settings.

Iterations:

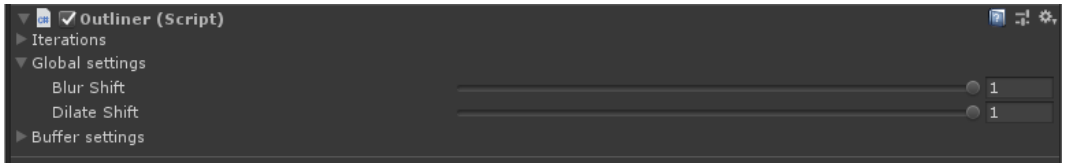


- **Blur iterations** – how many blur iterations should system perform when rendering outline (its performance heavy operation even though we optimized is as much as it possible, so *don't*

set to high values especially on mobile devices). The more iteration you set the softer outline you are able to render.

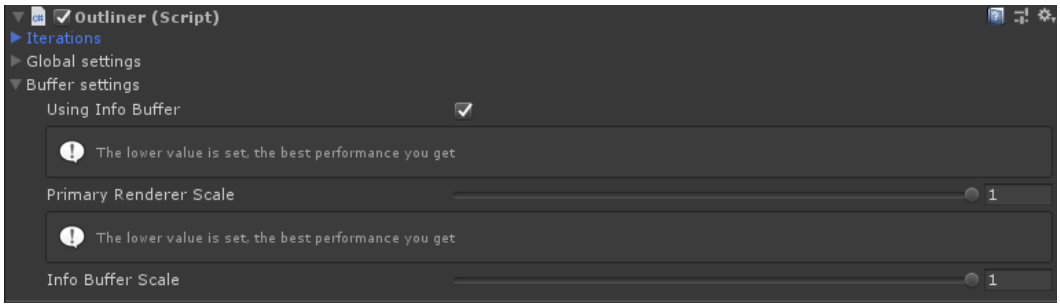
- **Dilate iterations** – how many dilation (extruding hard edge) iterations should system perform when rendering outline. It's less performance heavy than blur, but has its cost, so be mindful when changing the setting. The setting is perfect for getting rough commix-like edges.

Global settings:



- **Blur shift** – scale of the blur kernel. The smaller the value, the rougher edge you will get and slimmer outline will be.
- **Dilate shift** – controls the thickness of dilated part of the outline. The higher the value the thicker outline will be.

Buffer settings:



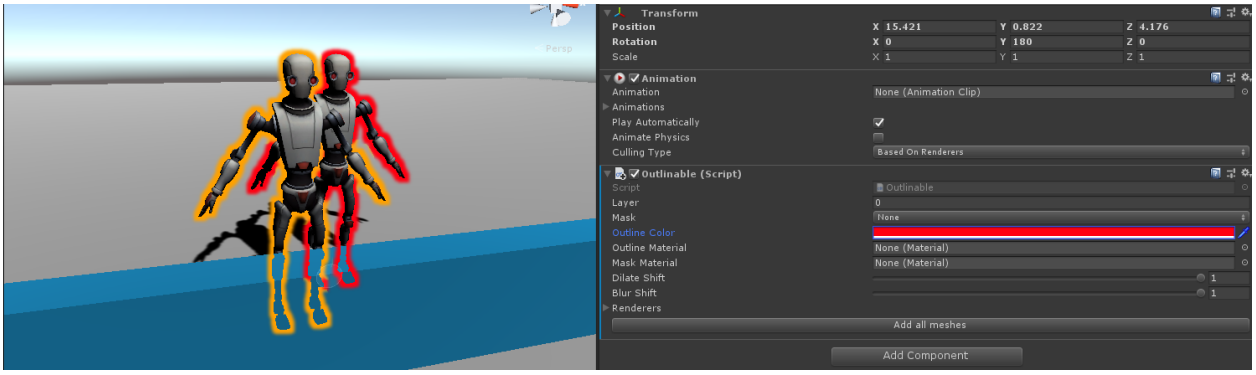
Buffer settings provide fine control over the surfaces that is used to render the image. Generally the lower scale of the renderer and info buffer – the better performance you get. Keep in mind it's necessary to get as lowest setting that fits your game style as possible (especially on mobile devices).

- **Using info buffer** – if enabled you can set per renderer settings on each game object that should be outlined. If not only global settings will effect rendering style. Has minor performance impact, but you can avoid using it if you stick to single rendering style of the outline for your game. In that case – just disable it.
- **Primary renderer scale** – how large the buffer for outline rendering should be. The smaller value you set – the lower quality you get but the better performance will be. The tradeoff should be mad for low performance devices such as mobile (Although its possible to run outline in native resolution, it's rarely required).
- **Info buffer scale** – how large the info buffer should be. In info buffer we keep info about how should each instance of the object render. It can be decreased greatly without losing too

much graphical fidelity. You should also set is as small as possible while keeping good image quality.

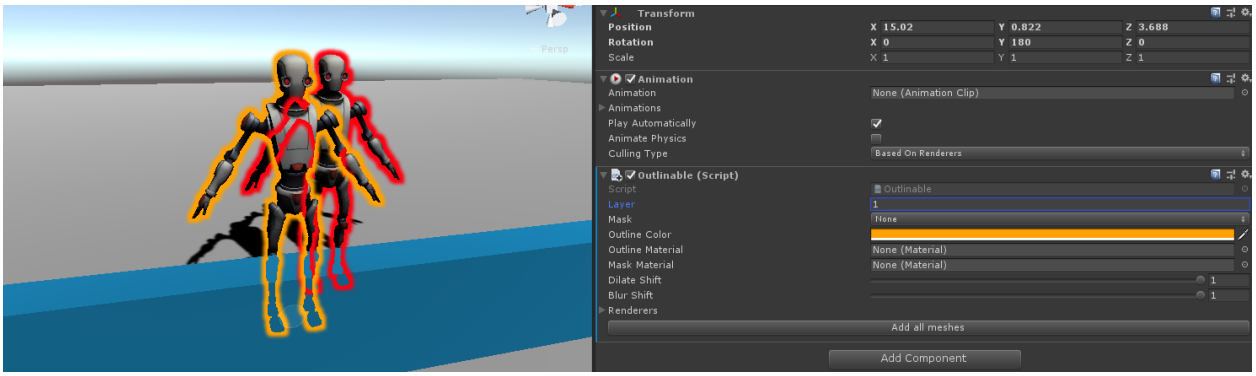
Outlinable:

The outlinable component provides rendering system info about which mesh should be outlined (supports every type of mesh renderer in unity) and provides per object settings for it (if info buffer is enabled).



- **Layer** – on which surface outline will be rendered. As you can see on the screenshot, the red outline is not rendered through already outlined robot in front. It's because of the approach of the rendering system. If you need to have some outlines rendering after one another, you can use different layers for them. Be aware, you should not use more than few layers because of the performance reasons. Most likely you wouldn't use these settings at all, but the possibility exists.

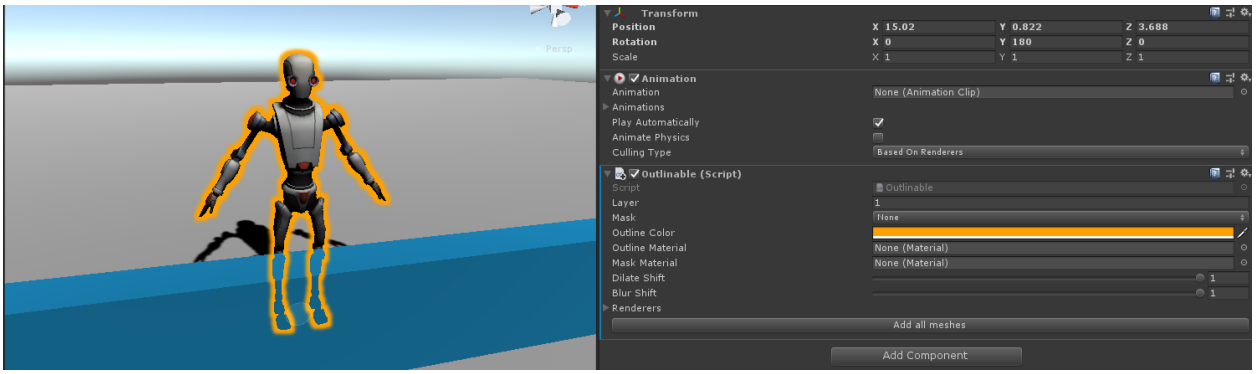
If we set different layers the result will be:



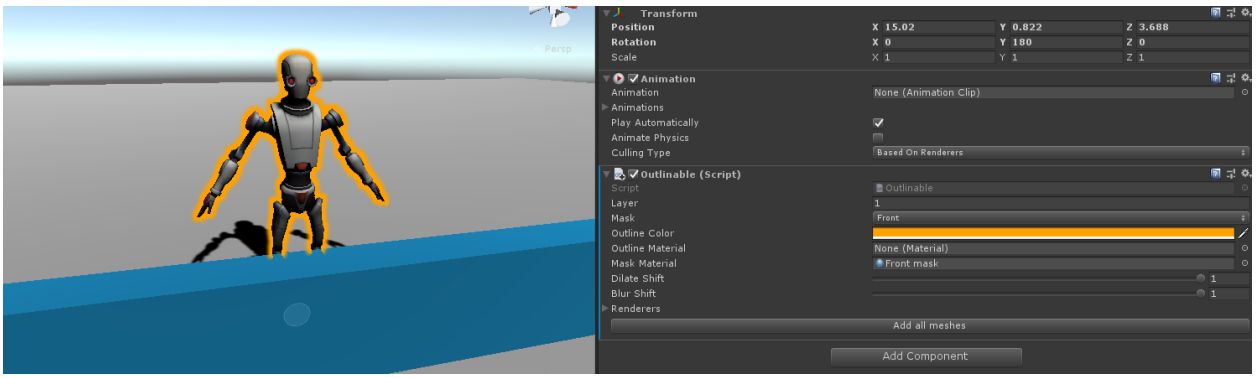
- **Mask** – the type of masking. You can choose what part of the mesh should be renderer. The whole mesh outline – None settings, only the part from behind of the obstacles – Back, and only the part that is visible – Front.

All three of the options:

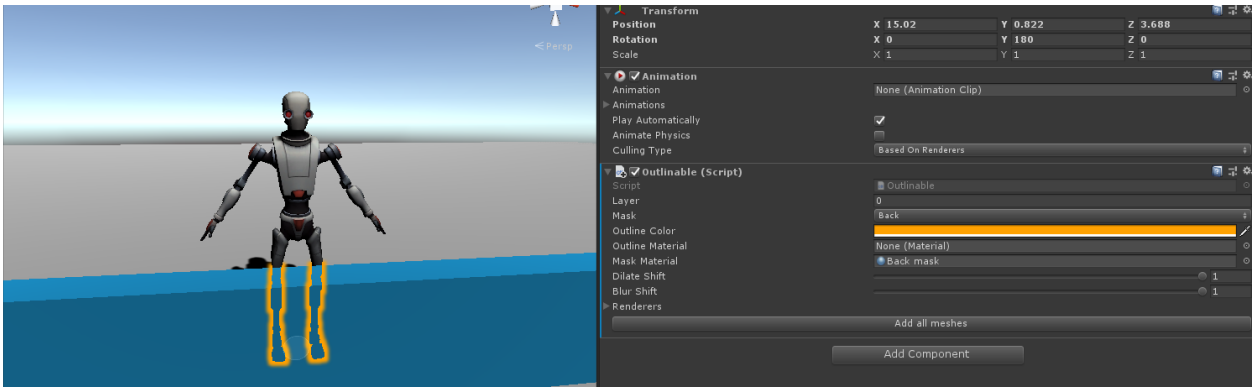
None



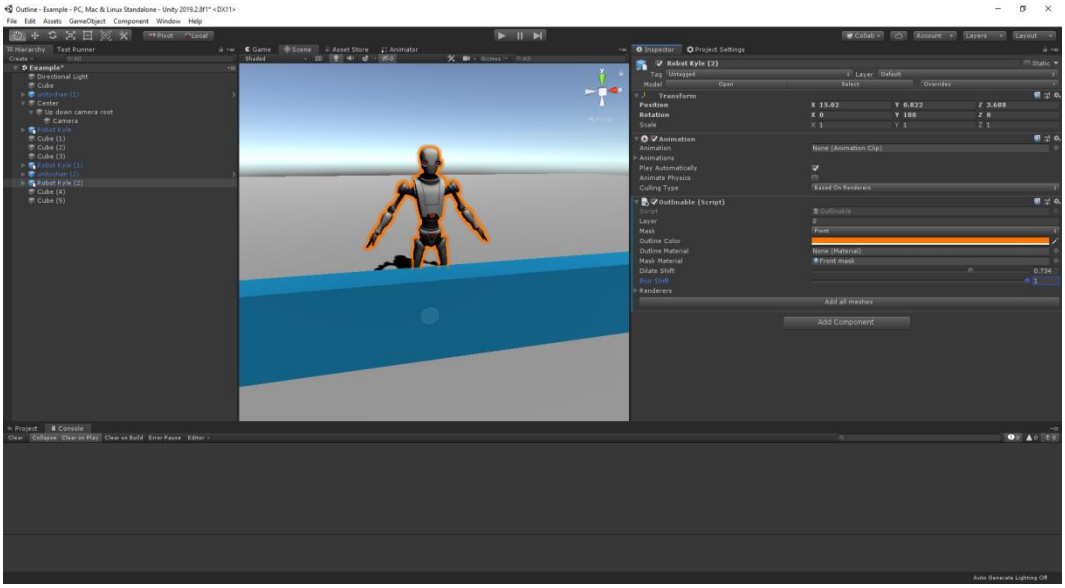
Front

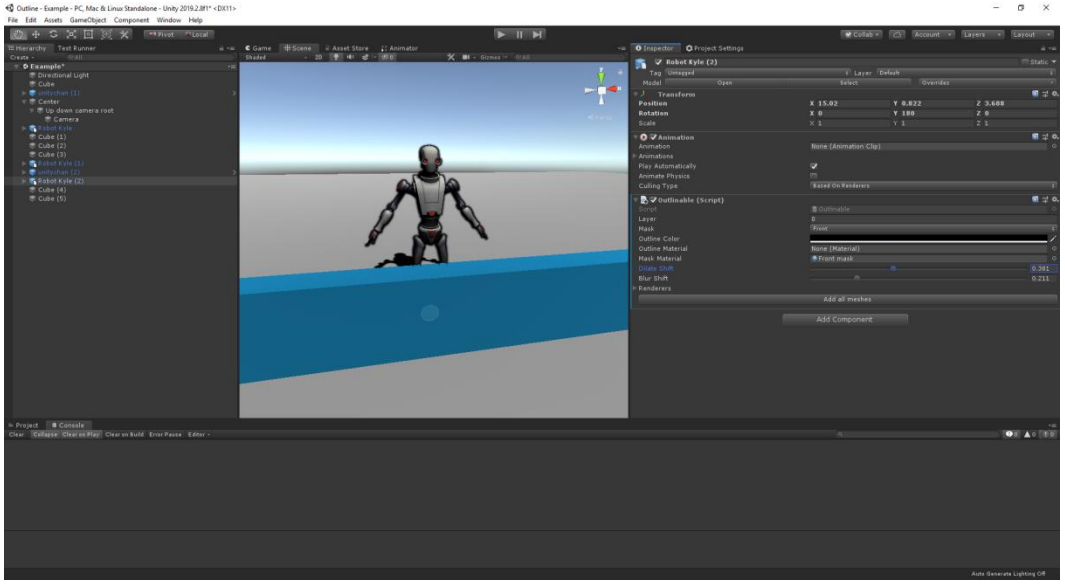


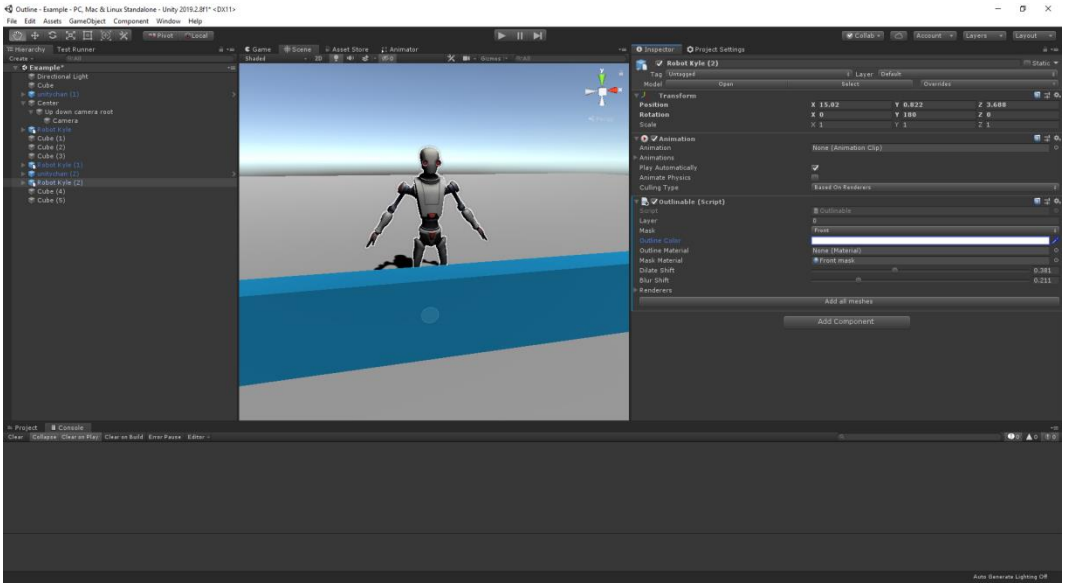
Back



- **Outline color** – is the color of the outline you are drawing. Even supports black, and white.

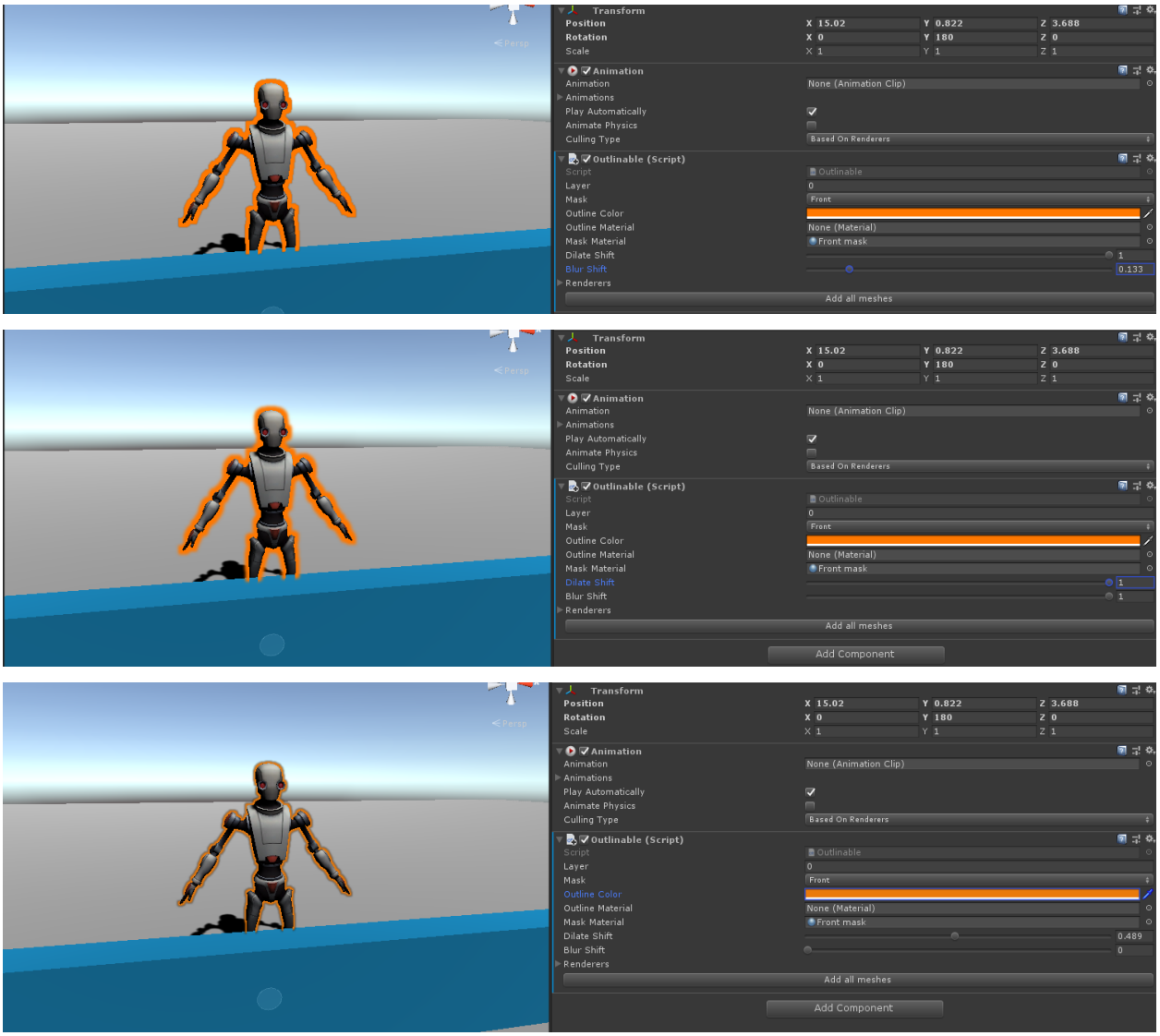






- **Dilate shift, Blur shift** – look at the Outliner component description.

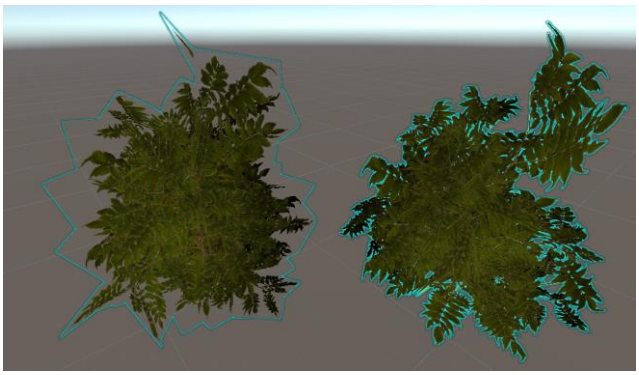
Examples:



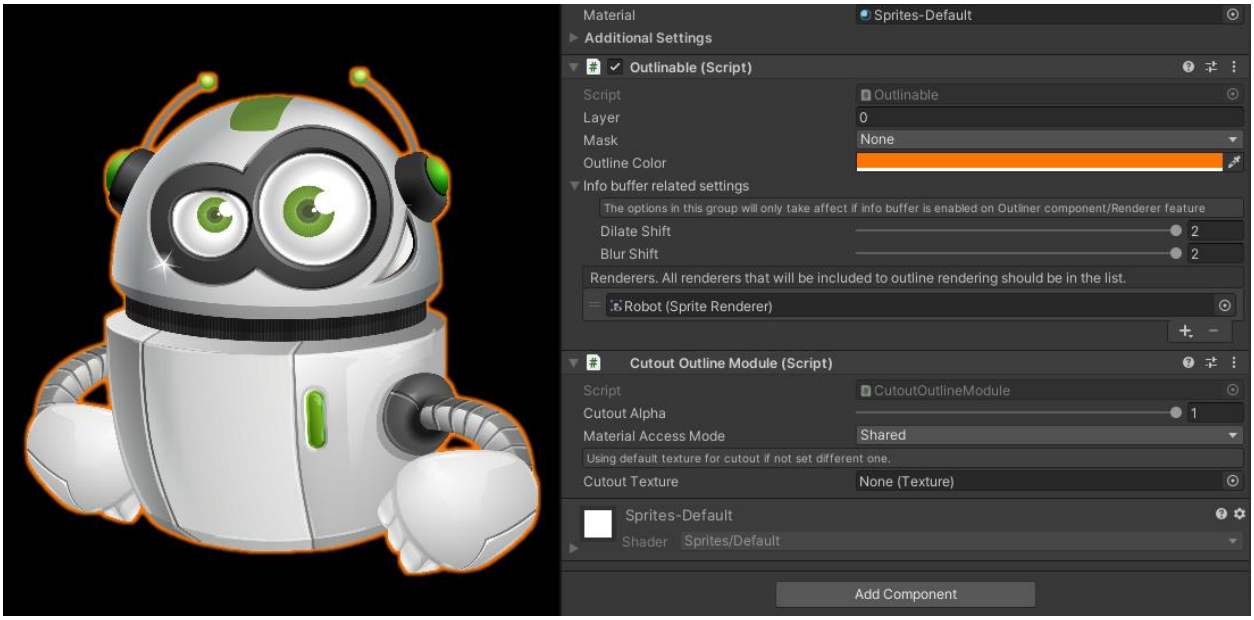
- **Renderers** – the list of renderers you want to be included in outline rendering. Can be automatically filled using button ‘Add all meshes’ on the bottom of the components inspector. It will get every Renderer in every child of the game object that Outlinable is attached to.

Cutout/Sprites support:

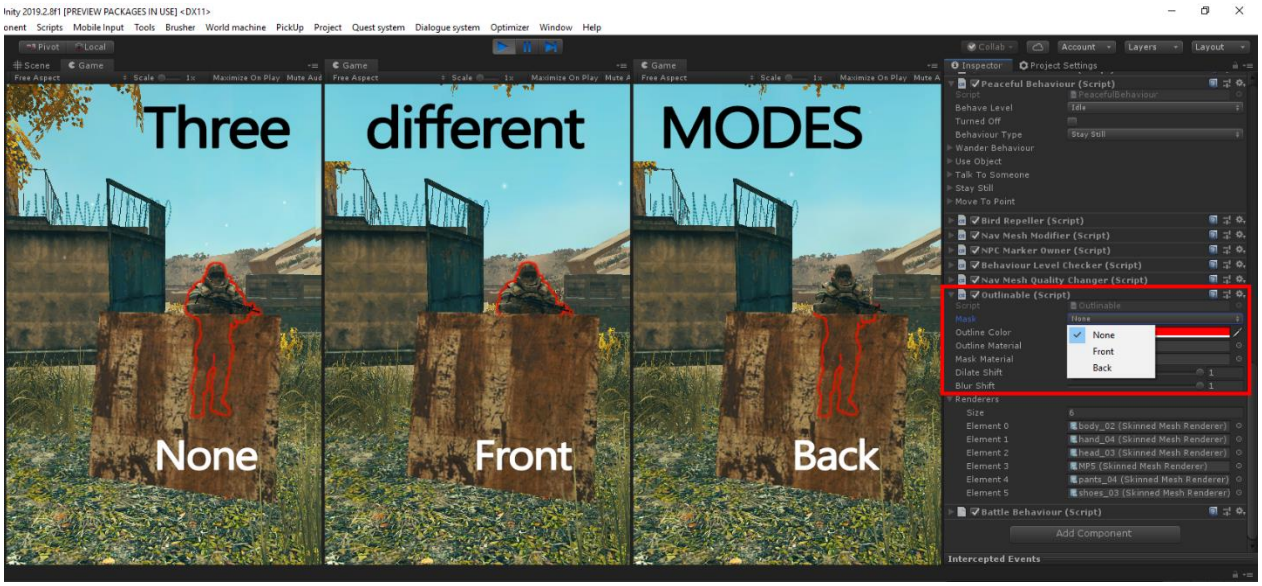
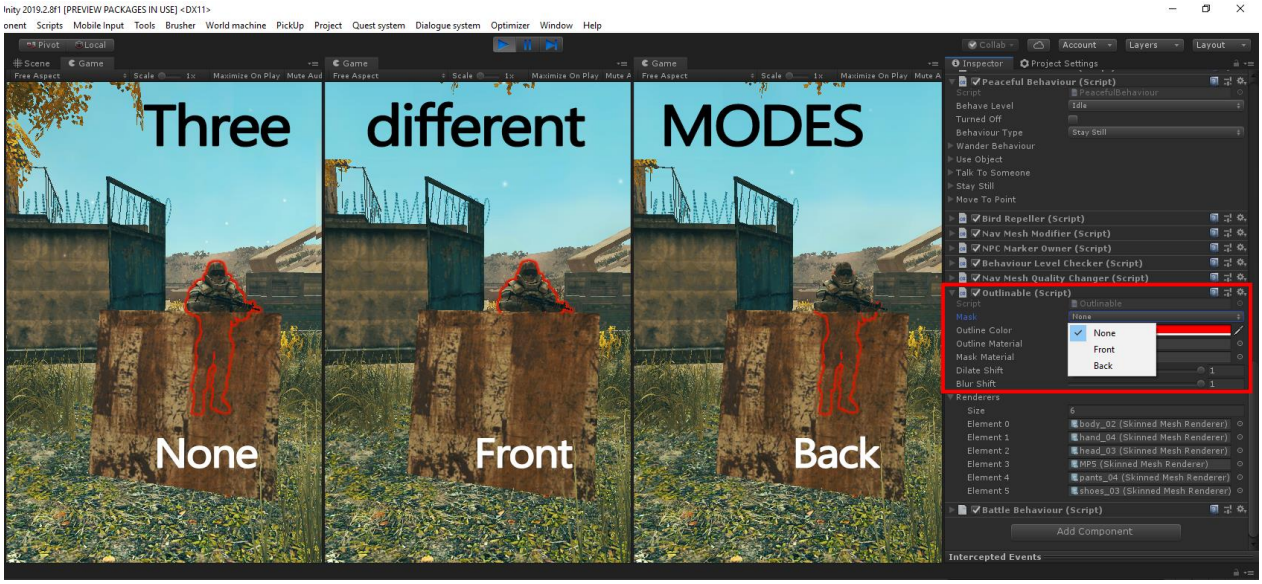
If you wisho to highlight your objects with cut out shader or sprites, you have to add CutoutOutlineModule component to renderers (not to the root object) that will be outlined with respect of your cutout texture.

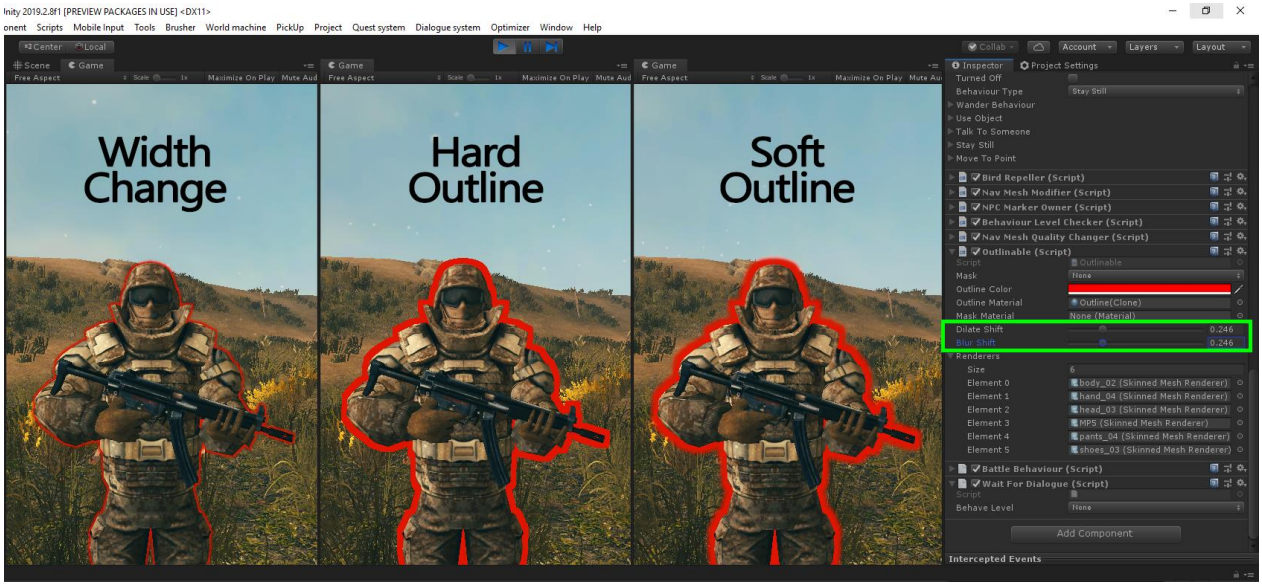


Example with no cutout module setted up:



- **Cutout alpha** – Controls cutout amount. The larger, the less transparent pixels will be included to the final outline.
- **Material access mode** – Controls wether outline will get *.material* property or *.sharedMaterial*. If your material do not change and you don't use instanced materials, leave it *Shared*. In other case you can set it to *Instance*.
- **Cutout texture** – In case you want to use custom texture for outline cutout, you can set it to the field. Otherwise it will get *.mainTexture* from material, or sprite texture in case of *SpriteRenderer*. Most of the time you can leave it intact.

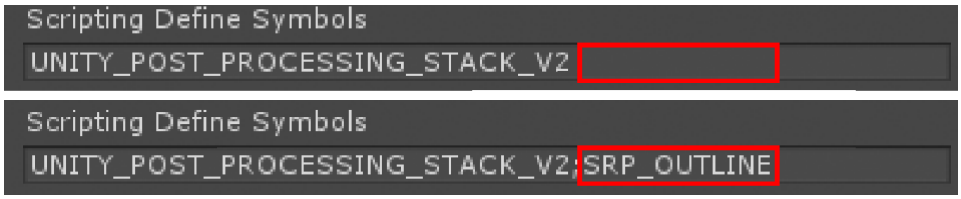
- 
Three
different
MODES
None
Front
Back
- 
Three
different
MODES
None
Front
Back
- 
Change
Outline
Color


Width Change
Hard Outline
Soft Outline
Inspector
Project Settings
Outlinable (Script)
Dilate Shift
0.246
Blur Shift
0.246
Add Component

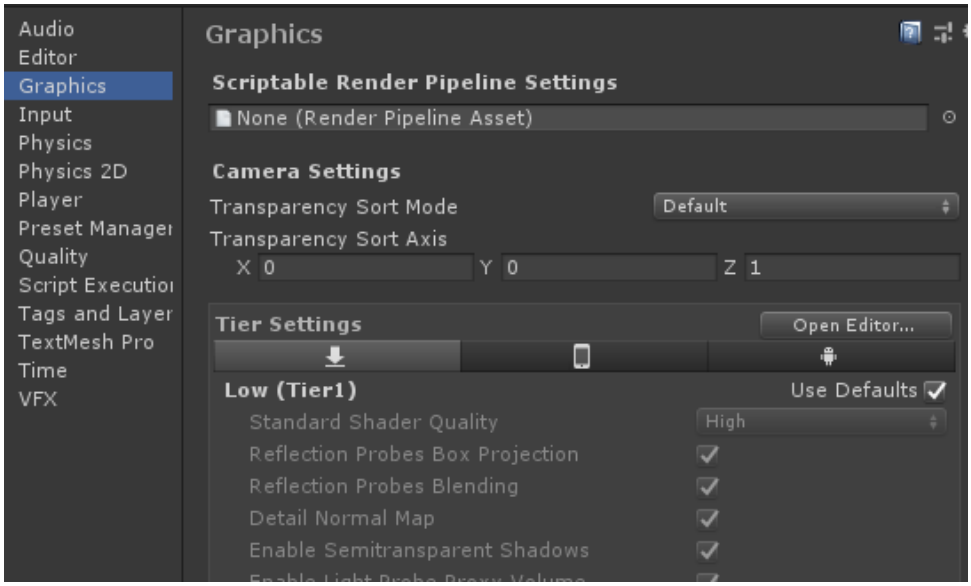
Using URP

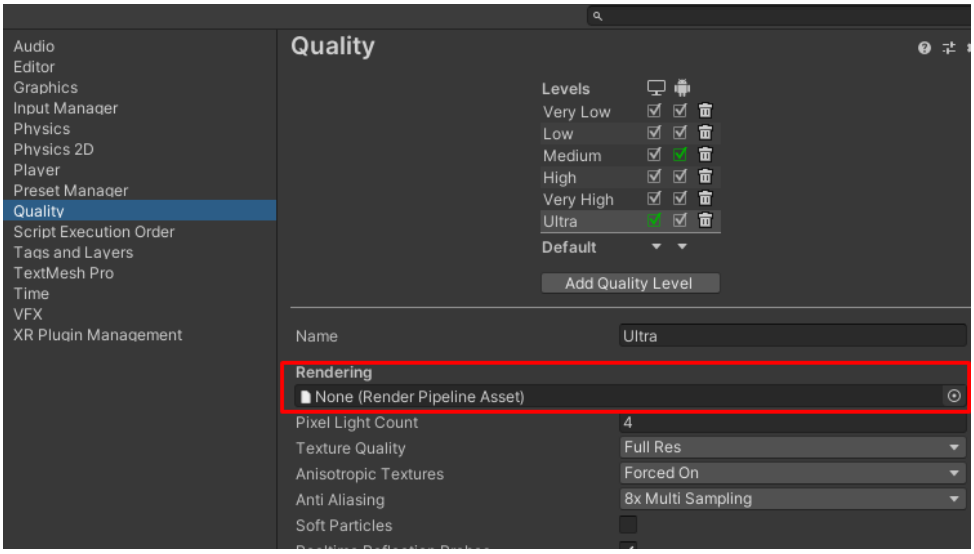
Warning! LWRP is supported since 2019.1. Earlier versions might not work due to API changes.

1. To use URP you have to import it from package manager
2. Add scripting define symbol: SRP_OUTLINE



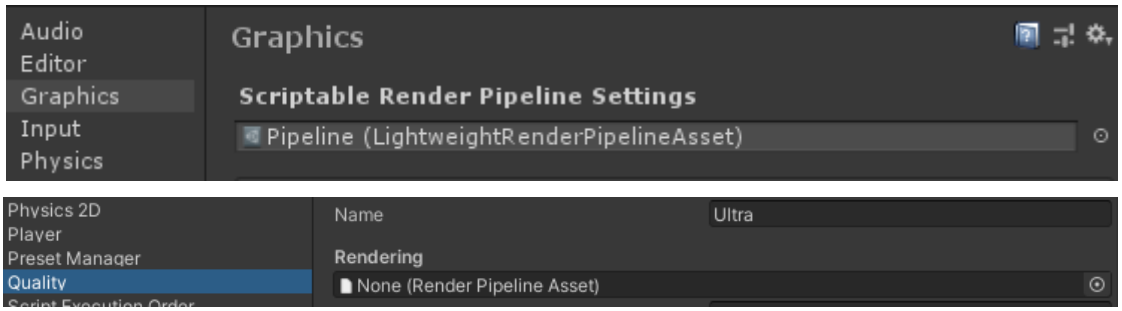
3. Open project settings (Edit->Project Settings) and select graphics and quality tab (please not that you have to change both):





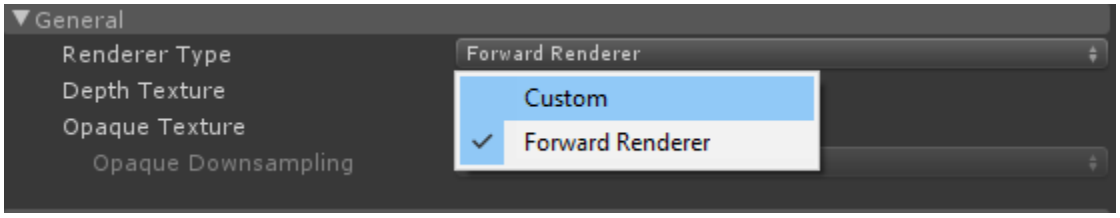
4. If you don't have existing render asset pipeline then create new one.

5. Assign newly created asset to slot in project settings window (Graphics and quality):

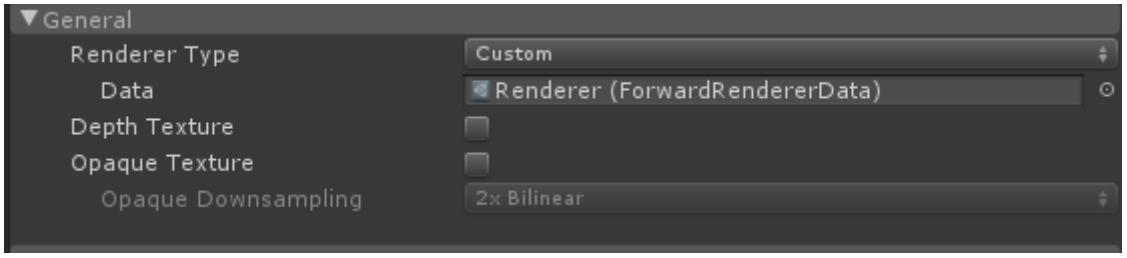


6. Create forward renderer:

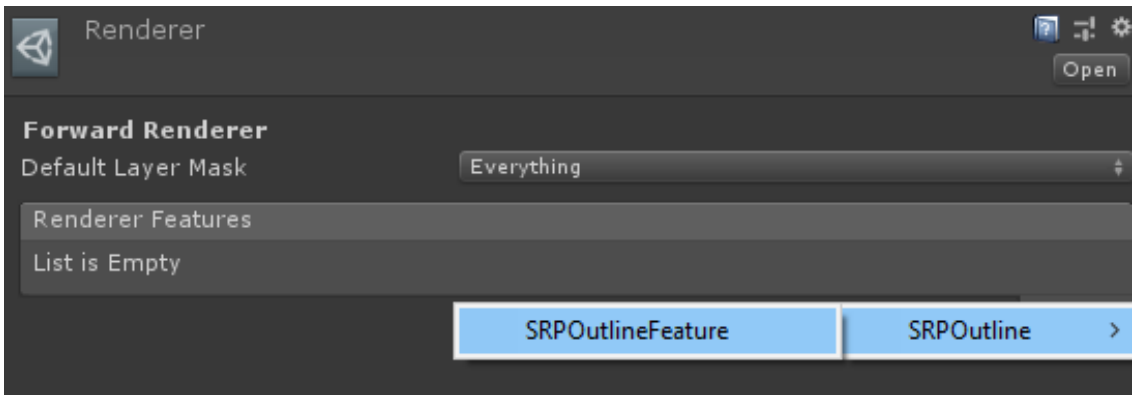
7. Select your pipeline asset (not renderer asset) and change **Renderer Type** property to *Custom*

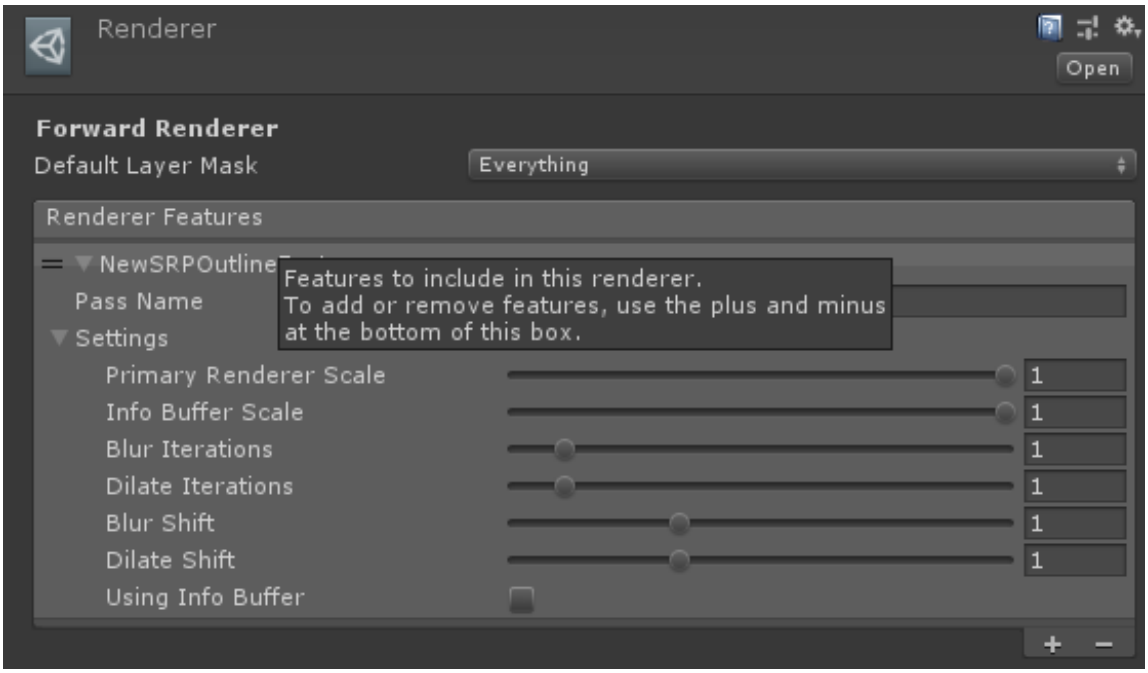


8. Assign your renderer to *Data* slot:



9. Add feature to your renderer:





10. Tweak the settings and everything should work out of the box.

Thank you for buying the Easy Performant Outline, we hope it will help you in your project.

If you liked our product, do not forget to rate it in an Asset store. We will be very grateful!

If you have any suggestions or questions, you can write to this email:
pirate.parrot.software@gmail.com