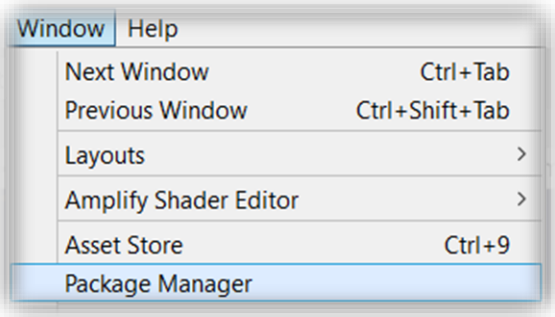
Thanks!

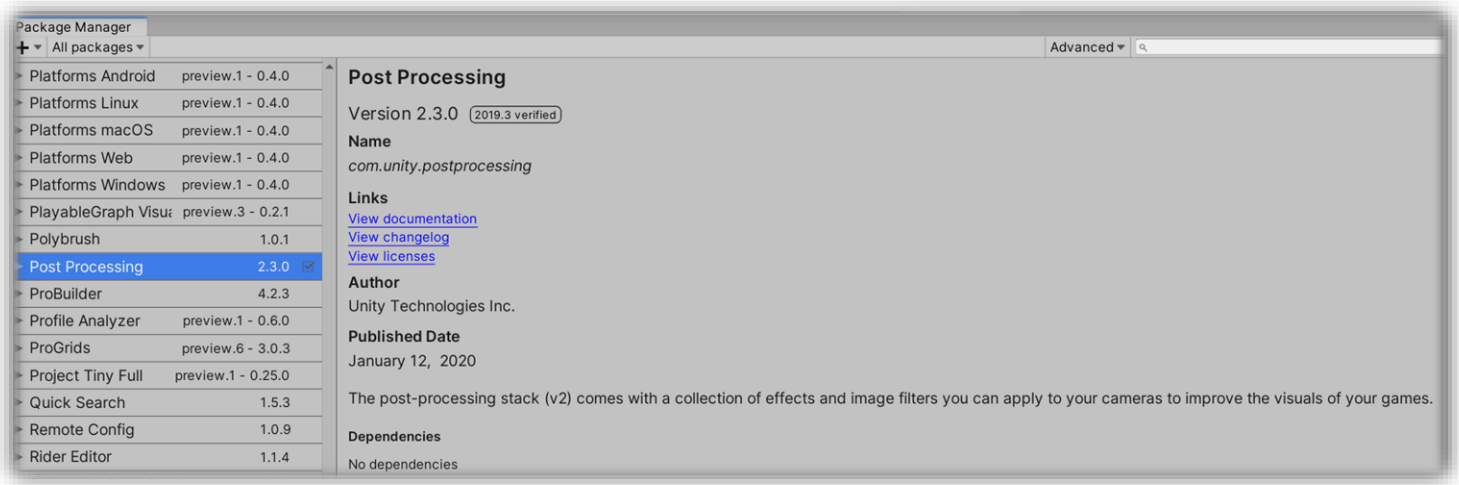
Congratulations and thanks for acquiring **Cyber Effects – Circuitry**!

Check for other awesome assets on my friend's page bit.ly/DeveloperPage-Elvis and know more about the developers behind the development of this visual effects series: twitter.com/ivangarciafilho and https://twitter.com/elvismdd.

Demo scene

The demo scene requires you to download and install Unity's Post processing stack from package manager, however, even with some missing scripts, It' should run, even without all the juiciness of the post effects!

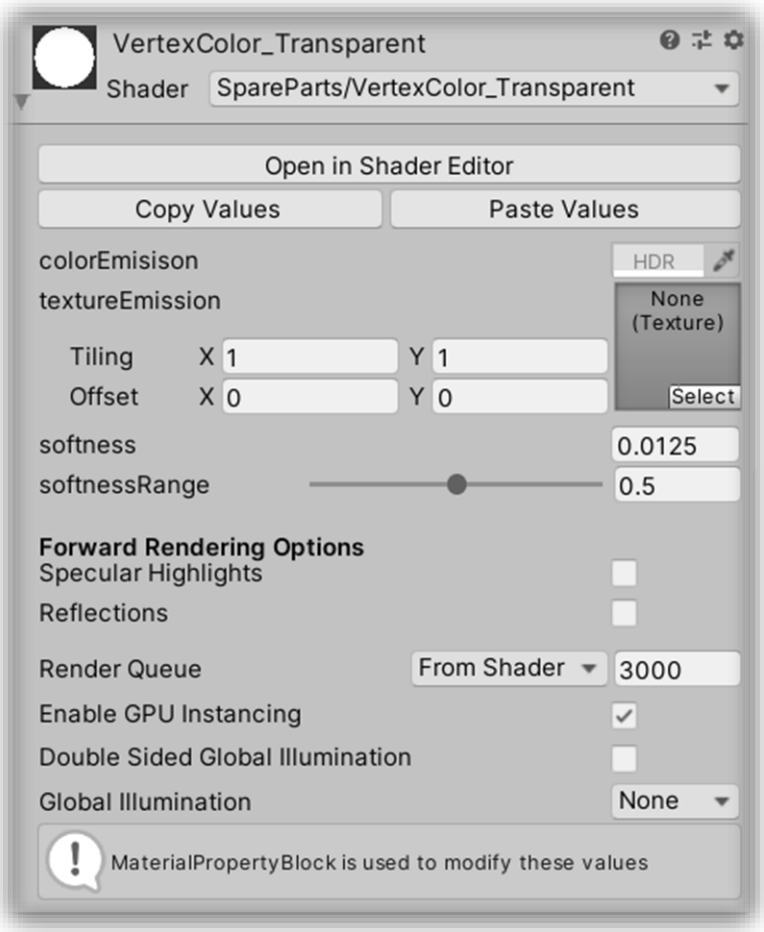




Shaders

Simple Unlit shaders that I added some brightness control through particle system's custom vertex streams, this way, I could use 1 single material for everything and then manage to make the black plating emissive or not, so the bloom post effect would capture their color brightness exceeding or not the threshold.

The **VertexColor_Transparent** is an alpha blended unlit shader and the only difference to other variants (multiplicative or multiplicativex2) uses different blending techniques.



colorEmission multiplies the particle color by chosen color and the texture does the same, injecting that value straight into the emission channel.

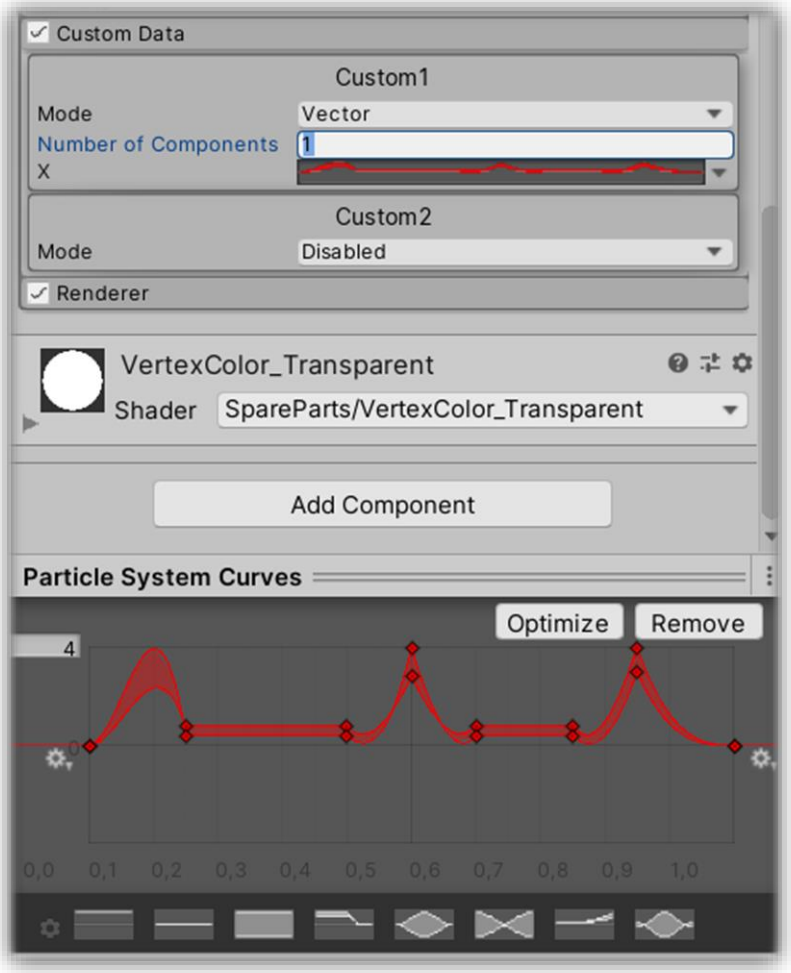
softness and **softnessRange** smooths the intersections between particle and geometry.

Those shaders are fully compatible with **Amplify shader** and if you have your own shader, you should just read the incoming vertex **TEXTCOORD0.w** and use the input value to multiply the color that goes into the emission channel.

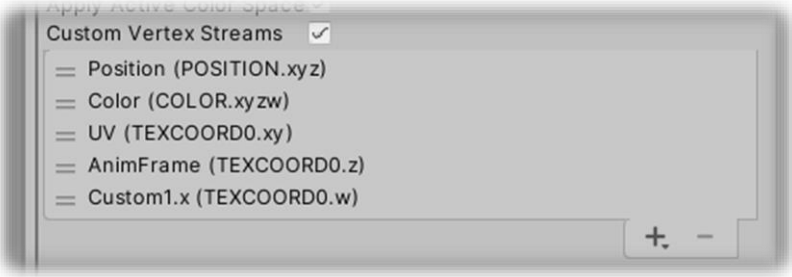
Custom Vertex Streams

Every **Render_** particle is sending to the shader through the **custom data** module a float parameter generated by an animation curve evaluated over its lifetime.

Allowing the particle system to handle the brightness of the emission color.



On **renderer** module, the **custom vertex stream** is toggled on and may contain some useless parameters being send towards the shader, just because I had to make all particles use exactly the same **TEXCOORD.W**, otherwise, my shader would not read the right parameter when multiplying the emission color.



That's a bit weird to explain, however, for amplify shader, the UV CHANNEL 1 is the UVCHANNNEL 0 of the vertex of the particle, they start counting from 1 instead of 0.

Then since **UVCHANNEL = TEXCOORD**, thus **TEXCOORD.W = UV.T** within amplify shader.