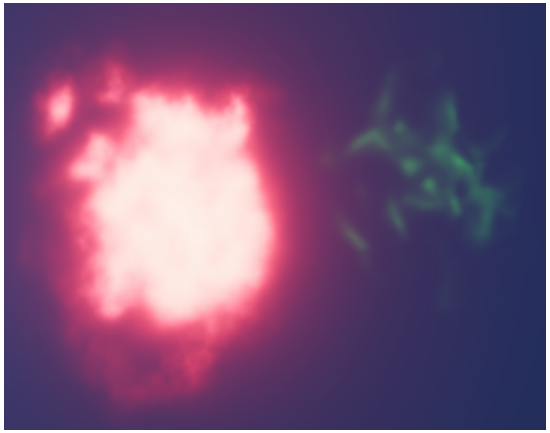
Unity Shader Graph Workshop
Part 3: Custom Shaders + Particle Systems

Dr. Micah Schuster

Summer 2021

School of Computing and Data Science

Wentworth Institute of Technology



Wentworth INSTITUTE OF TECHNOLOGY

Particle Systems?

Our Diablo 3 style shader is best used in a particle system.

However, there are some major issues if we apply what we've created to our particle material:

- Color over lifetime doesn't do anything
- All scrolling starts in the same place for every particle (no randomness per particle)

Wentworth INSTITUTE OF TECHNOLOGY

Particle Systems: Brief Introduction

- Particle systems are typically used as visual effects representing everything from fire and smoke to sparks to weapon fire.

There are many tutorials out there for creating specific particle systems.

Today, we'll create a basic system and then reproduce a partial effect from a game.

Wentworth INSTITUTE OF TECHNOLOGY

Particle Systems: Brief Introduction

Particle systems are just a component that is added to a GameObject.

In each particle system there is an initial area for settings. Below that are other modules that can be activated to provide different options for how the particles behave.

You'll usually always use "Emission", "Shape" and "Renderer". The others will depend on the situation.

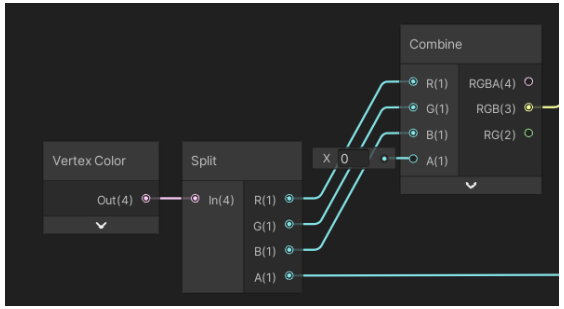
The Renderer module is where we will apply are material to the particles.

Let's play around with a particle system first before we jump back to our shaders

Vertex Color

When using the color over lifetime property of particle systems, the color (and alpha) is applied to the vertices of the particle quad.

Shader Graph can take that information, using the *Vertex Color* node, and use it in the shader.



Note that I split the vertex color into its RGBA components.

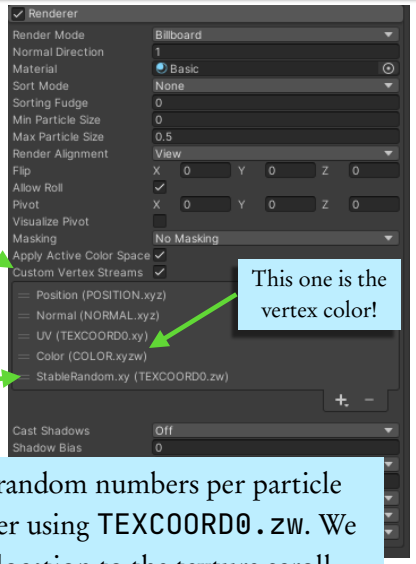
I recombine the RGB to multiply with the color of the texture.

You should do the same with the separate alpha channel, but multiplied with the alpha from the textures.

Better Randomness: Custom Vertex Streams

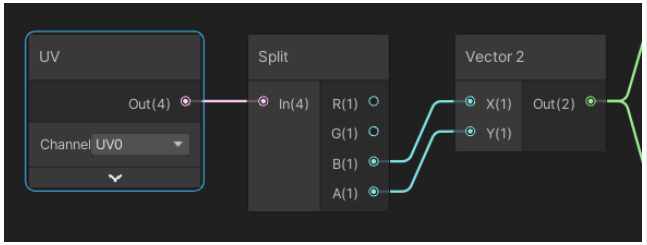
By default, custom vertex streams are not enabled in you particle systems. You must click the check box in the Renderer component:

From here, you can add additional information that can be sent to the shader, click the + icon and select `StableRandom.xy` so that it appears in the list.



`StableRandom.xy` generates two random numbers per particle (`0.0-1.0`) that are sent into the shader using `TEXCOORD0.zw`. We will use these to give a random start location to the texture scroll.

Better Randomness: Shader Input



`TEXCOORD0` is the UV0 channel in the UV node.

Since the two random numbers are `zw`, they correspond to Blue and Alpha if we do a split.

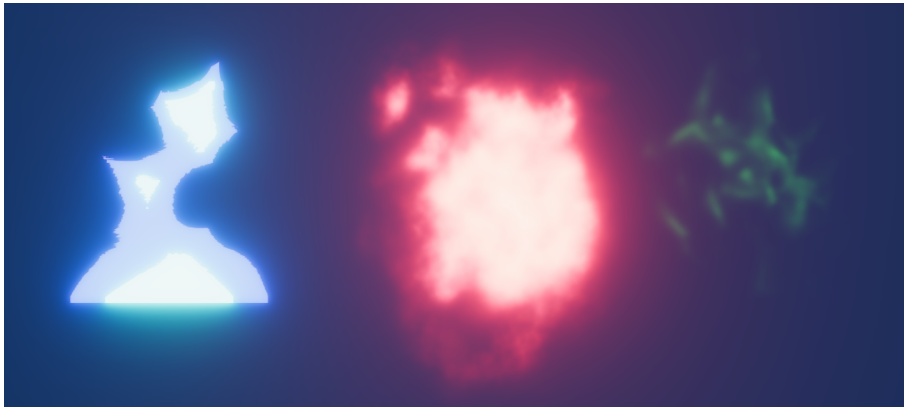
RG are the `xy` UV coords on the particle.

Here, I just put them into a Vector2 and add them to the offset value before it gets input into the Tiling and Offset node.

Now, each particle will have a slightly different starting place for the offset of the RGB and Alpha textures.

Final Scene

In the starter repo, I've included a Main scene that includes three effects. The RIME fire and two effects using the Diablo style shader.



Wrap Up

Take what you've seen over this three part series and create cool new shaders and effects.

Wentworth
INSTITUTE OF TECHNOLOGY