Unity Shader Graph Workshop
Part 1: RIME Stylized Fire

Dr. Micah Schuster

Summer 2021

School of Computing and Data Science

Wentworth Institute of Technology

Wentworth INSTITUTE OF TECHNOLOGY

Shaders

While shaders are a cool topic, a full intro is way beyond the scope of this workshop. We could have several semesters dedicated to shaders and learning a shader language.

Fundamentally, shaders are programs that *shade* 3D scenes.

They provide the appropriate levels of light, dark, and color to materials in the scene. These days, they also help with special effects and can be used as computational programs as well.

In essence, when you write a shader program, you are programing on the GPU.

Unity provides a few useful shaders for doing standard things, but for anything beyond that typically requires a custom shader.

Wentworth INSTITUTE OF TECHNOLOGY

Shader Graph: Getting Started

Make sure you have:

- At least Unity 2019 or later
- HDRP or URP

The starter project uses Unity 2020.3.0f1 and URP

Ensure that Shader Graph package is loaded via the package manager.

Create a new Shader Graph:

Create → Shader → Unlit Graph

Typically, I create a **Shaders** folder to store them all in.

We will also need a material that the shader is applied to, right click on the shader: Create → Material (and give it a name)

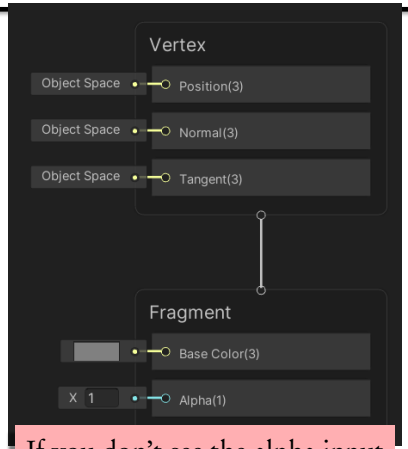
Wentworth INSTITUTE OF TECHNOLOGY

Shader Graph

Here is our Master node

It contains two parts, the vertex shader and fragment shader.

We will only be using the fragment part: color and alpha



Each of the little circles on the left are inputs, and the boxes on the left are "default" values. We can either change them or hook other nodes into those values.

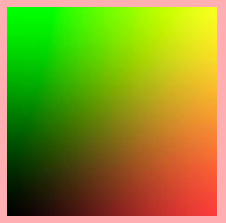
If you don't see the alpha input in the fragment shader, you'll need to set the shader to "Transparent" in the global properties.

UV Distortion

UV coordinates are typically used to map a 2D image unto a surface.

- When you bring a texture into Unity, that texture is mapped to a square in UV space.
- Any distortion to the texture's UV coordinates will distort the texture.
- This is the basic concept for the shaders we will create throughout this workshop.

UV coordinates are usually represents by a 2D colored square:



The colors are (Red, Green) mapped to (x,y) on the image.

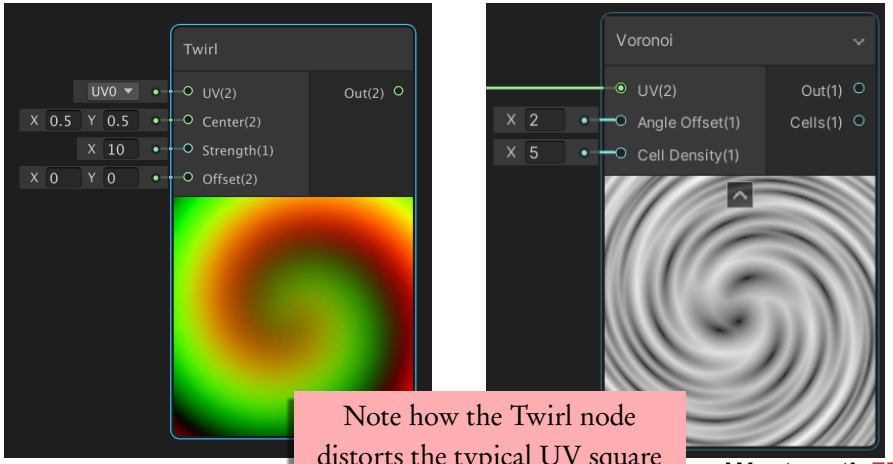
For example, (0,0) is the lower left, black, and (1,1) is the upper right, yellow.

Wentworth INSTITUTE OF TECHNOLOGY

UV Distortion Example

The Twirl node skews any UV coordinate into a spiral like pattern.

Hook the output of the spiral node into the input UV of a texture:



Note how the Twirl node distorts the typical UV square

Wentworth INSTITUTE OF TECHNOLOGY

Time node for Scrolling Textures

Add a "Time" node to the graph.

This is a very useful node!

This allow us to change properties of the shader over time!

Using a Time node with a Tiling and Offset node allows us to "scroll"our UV coordinates, thus making our texture appear to scroll over time.

Scrolling textures are the key to many interesting effects

Shader Graph: Couple of things

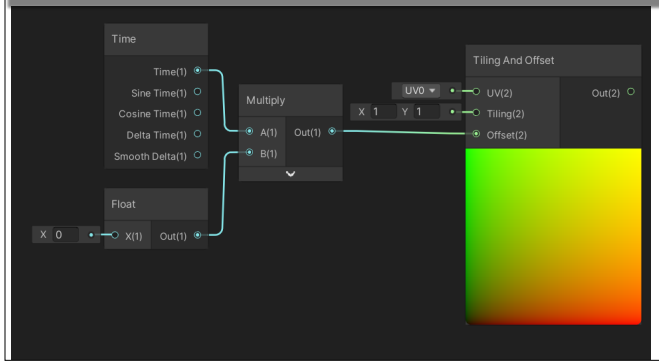
Here's a couple of useful things for dealing with Shader Graph:

- You can delete a connection by right clicking on it and selecting 'delete'
- You can delete nodes by right clicking on their top bar and selecting 'delete'
- Dragging and output line from a node to empty space will immediately allow you to create a new node
- The number (and color) next to the output/input represents the number of values associated with the output and the types of those values. Not all inputs can match to all outputs!
- An output can go to multiple inputs, but an input can only have one input

Basic Operations: Multiply

Let's control the speed of the scroll:

- Create a Float node and a multiply node
- Connect the output of the Time to the multiply and the output of the float to the multiply
- Connect the output of the multiply back to the Tiling and Offset node

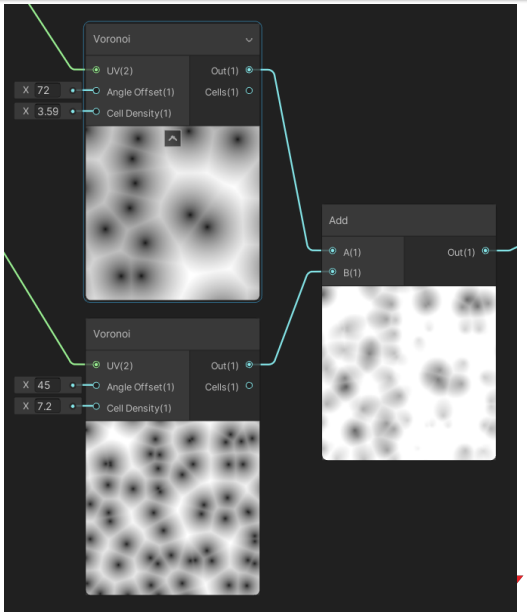


Adjusting the value of the float now changes the speed of the scroll

Wentworth INSTITUTE OF TECHNOLOGY

Basic Operations: Add

- For this shader, we'll use the sum of two noise textures that scroll at different speeds
- Adding and multiplying textures is very intuitive, the operation is done to each input channel. Use the preview to help get the effect you want.



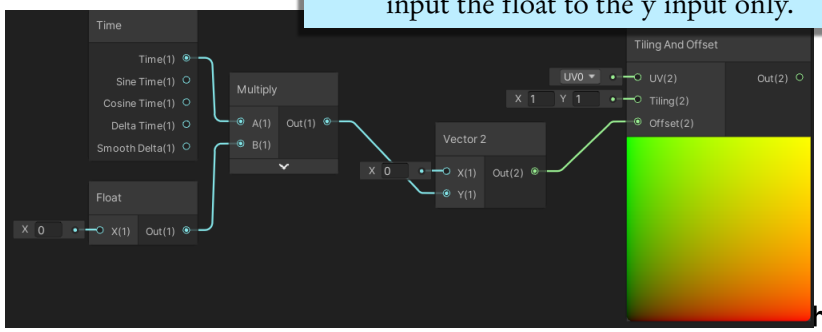
Input Differences

One thing you may notice is that our textures are scrolling at an angle.

The output of our Float is one value (blue), but it is plugged into a two value (green) input in the Tiling and Offset node.

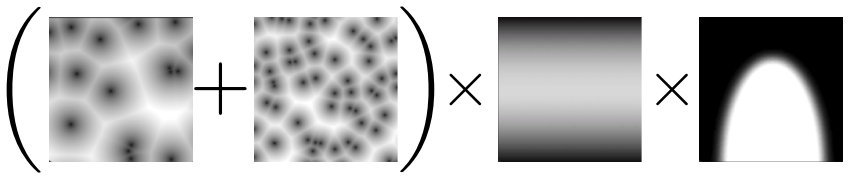
This applies the float to both inputs (x,y).

To change this, we'll use a "Vector 2" node and input the float to the y input only.



INSTITUTE OF TECHNOLOGY

Basic Shader Outline



Noise 1 Noise 2 Grad Alpha

The resulting texture is then added to the RGB texture UV:

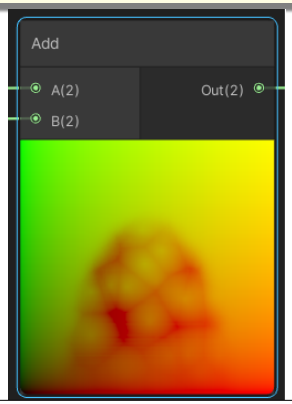
RGB UV

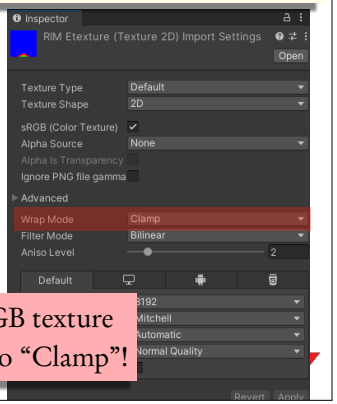
Wentworth INSTITUTE OF TECHNOLOGY

What Does This Actually Do?

The RGB texture UVs are changed based (mostly) on the noise texture movement. Thus, different parts of the RGB image are shown in different places (this is the UV distortion).

Currently, the UVs are lighter color in the distorted area, so we need to multiply by -1 before we add to the RGB UVs.



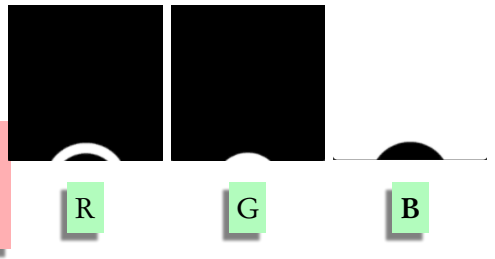


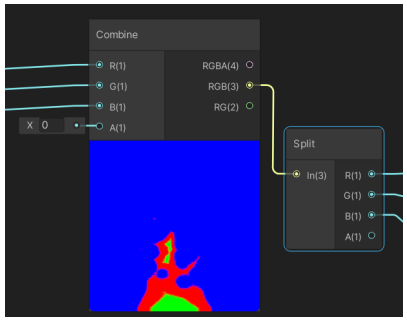
Note: Your RGB texture needs to be set to “Clamp”!

Texture Packing

While a rainbow fire might be appropriate for some games, let’s adjust the colors

Here, I’ve packed separate grayscale information into the same texture using the RGB channels



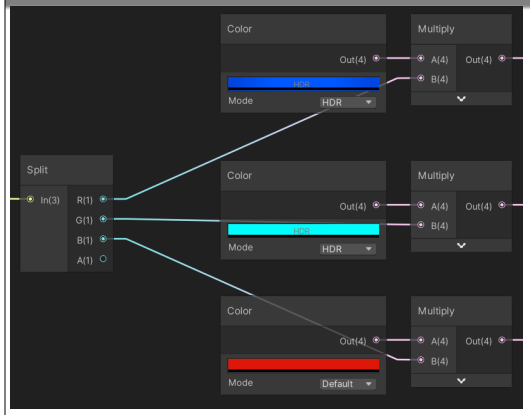


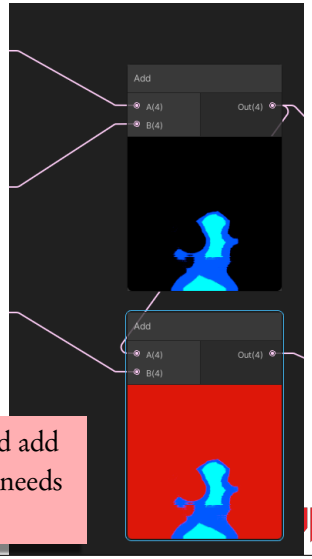
Using a Split node, we can separate each component to use independently.

Wentworth INSTITUTE OF TECHNOLOGY

Adding Color

Create a Color node and multiply it to one of the split RGB channels.



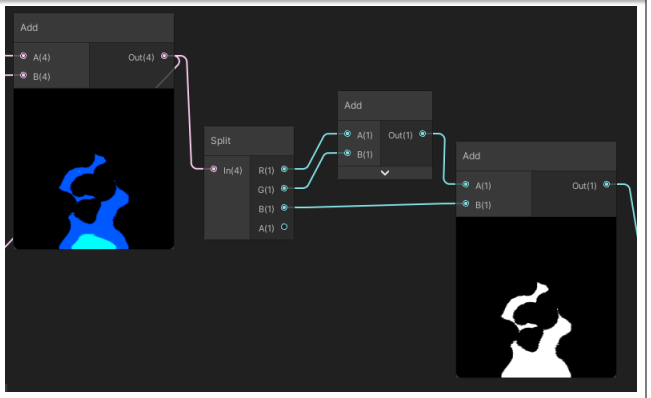


Do this for each channel and add them all back together (this needs two Add nodes)

Adding Alpha

For transparency, we need to do two things:

- Make sure our shader is able to be transparent
- Create an alpha “texture” to use in the alpha input of the Fragment shader



I create the input to the alpha by adding all the color channels from the “texture” created from the R+G channels.

Wentworth INSTITUTE OF TECHNOLOGY

Final Scene

I've created a couple of assets in Blender to use with the stylized fire, feel free to mess around with them:

Coming up:

Part 2:

- Recreate the base two-texture shader from Julian Love's "VFX of Diablo" GDC presentation

Part 3:

- Adjust our shaders to be useful in particle systems