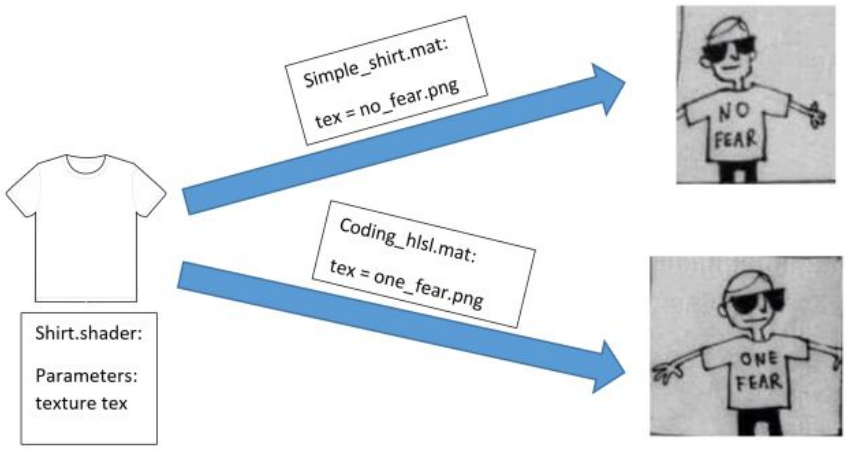
CSE 490j
Unity Introduction to Shader Graph

What are Shaders?

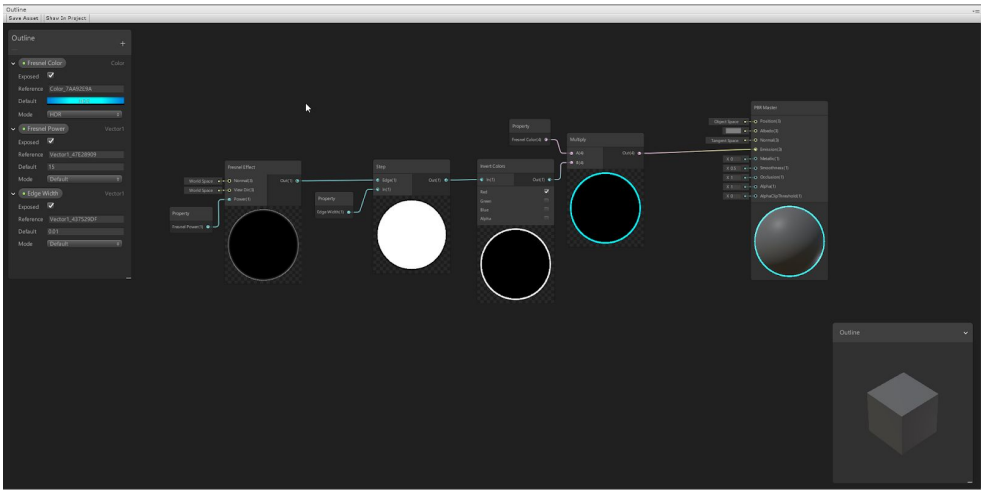
Shaders are scripts that control how an object is rendered on the screen and are how we control settings such as color, texture, and shininess. Materials take a shader and modify it's base settings. Each material can only have one shader, but a shader can be shared between multiple materials.

In a way, the shader acts as a template with parameters, while a material is just an instance of a shader with specific values assigned to those parameters.



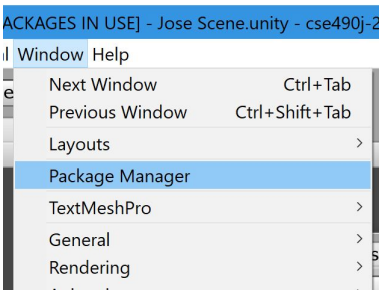
Intro to Shadergraph

Typically, you have to code shaders by hand. However, Unity has a feature called ShaderGraph that lets you create shaders through a node based system and is a lot easier to get started with.

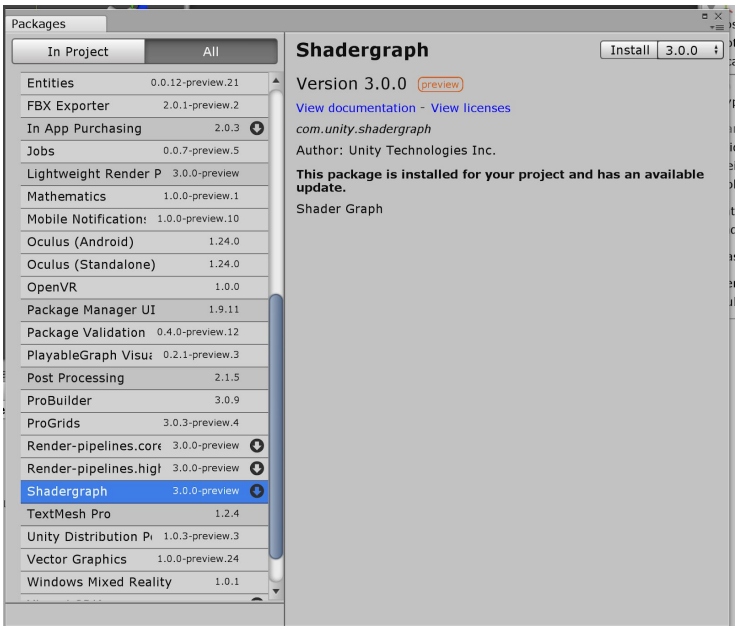


The first thing we have to do if install shadergraph.

- Go to window>Package Manager



- Click on All
- Scroll down until you find a package called “Shdergraph”
- In the top right corner, hit install



Note: if errors start to appear, objects start turning pink, or something else out of the ordinary happens, let a TA know. This may be an issue with shadergraph versioning.

Writing your First shader

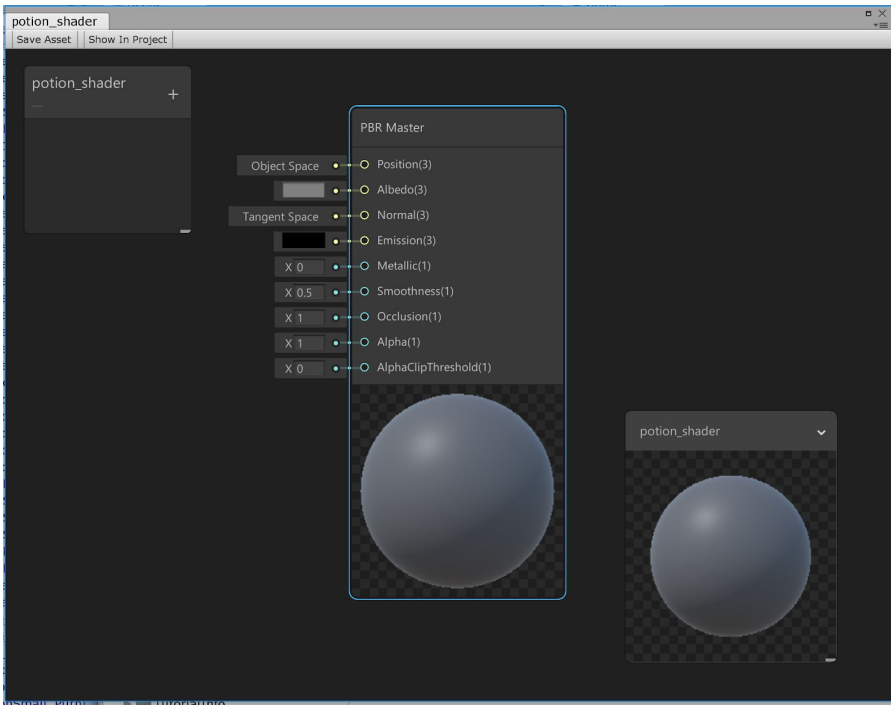
The first shader we will write is to control the look of the potion mesh.

- in your project tab, right click>create>shaders>PBR Graph
- Name it “potion_shader”
- Right click the shader

- create>material and name it “potion_mat”.
    - This automatically creates a material with this shader as a setting.

- Make sure you have the potion prefab in your scene (it should be in Low Poly Dungeon Assets>Prefabs>Props)
- Drag the new material onto the potion in your scene.

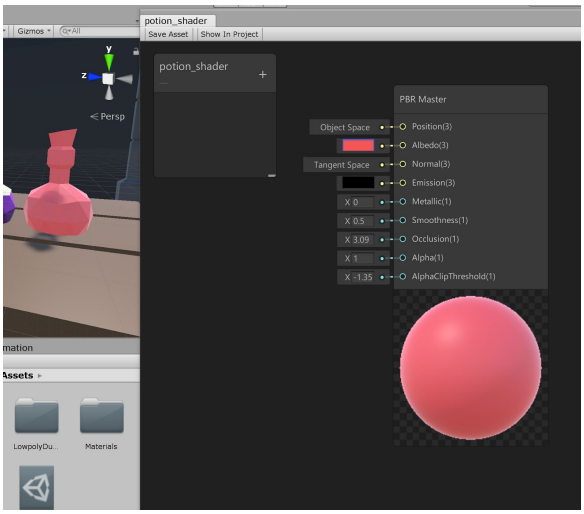
If you did this correctly, your potion should appear grey. This is because we haven’t set it up yet. So let’s do that. Double click “potion_shader” in the project tab to open up shadergraph. You should see something like this.



The bottom right box is a preview of what your final shader will look like.

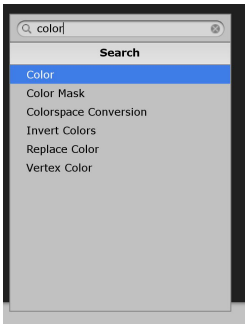
In box to the left are where you will see your custom parameters (more on that later).

The box in the middle is the Master node. You can see that there are number of different settings with values attached. Feel free to mess around with them and see how they affect the shader preview. Ty specifically to change the Color of our shader using the albedo. When your done, click on “Save Asset” In. the top right corner, If you go back to your scene view, you should see that your potion should have changed based on the settings you modified in shadergraph.

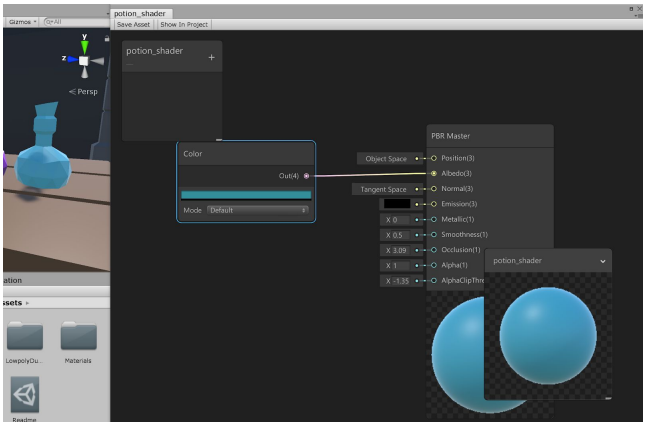


Now let’s start making nodes. Nodes are how you control and modify the look of your material. For example, let’s start with the color node.

- Hit spacebar, select “create node:” and then a node menu will pop up
- In the search bar type “color”



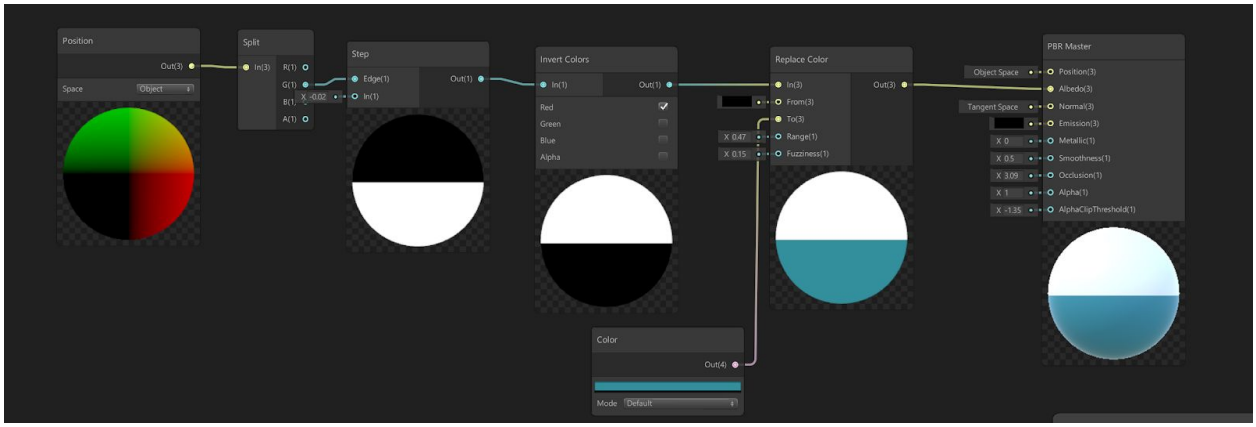
- Select the color node
- In the pink circle to the right, marked “Out(4)” click on it and drag into the yellow circle labeled “Albedo.
- In the Color node, click the black bar, and this will allow you to change the color of the node.
- In the preview, you should see the color of the shader changing.



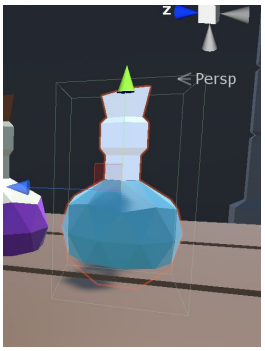
Now we are going to add a setting so that we can fill up the bottle with a potion. To do this, we are going to add some more nodes.

- Find and create a node called “Position” (the same way you set the Color node). Set its space to “Object”. This node represents the xyz coordinates of every pixel on the object.
- Create a Split node and connect the Position’s Out(3) node to the Splits In(3). This node will split our position into separate x y and z values.
- Create a step node, and connect Splits G(1) node into the Step. If you mess around with it’s In(3) value, you will see that it will fill the preview circle different amounts vertically.
- Create an invert colors node and connect Step to its In(1). Hit the checkmark next to red, you will see that it flips the colors.
- Create a replace colors node and connect invert colors to its In(3). Take the Color node from earlier, and connect it to Replace Color’s To setting. You will now see that color in the preview.
- Pipe Replace Colors Out(3) field into the PBR Master’s “abledo” field.

If you set it up correctly, your graph should look something like this.

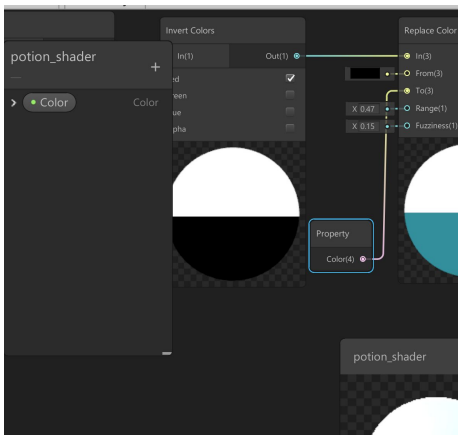


And if you save out your graph, the potion in the preview will look something like like this.



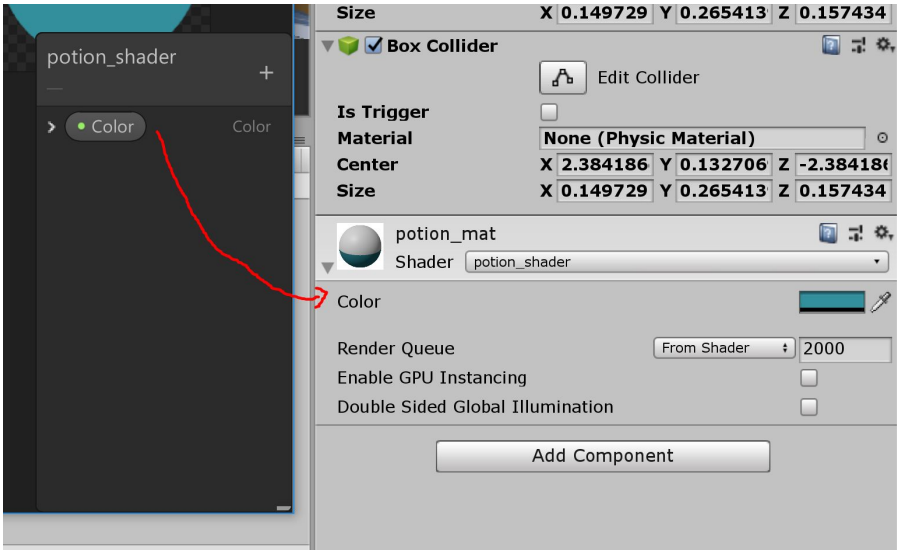
Now that we have the functionality of the graph down, let’s make it so that we can easily change it’s values.

- Right click the Color Node, and hit convert to property.
- In the box to the left, you should see a new box that says color



- Double click, and name it “fill color”

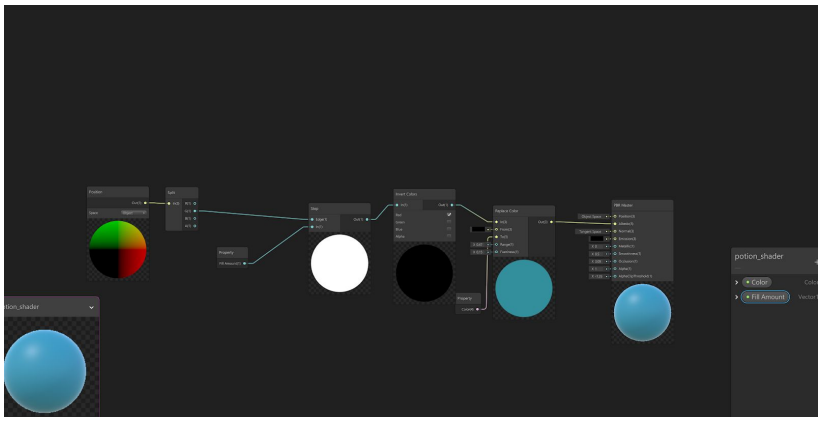
Making certain nodes into properties does a couple things for you, First it allows you to change these settings outside of shadergraph directly in the material settings itself. Secondly, it allows you to have different materials with these values linked to different settings.



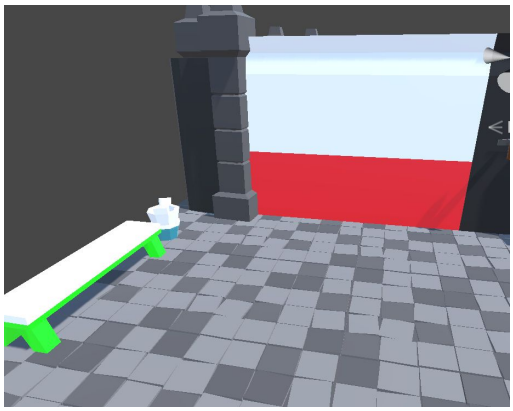
Let’s do something similar for the fill amount.

- Create a node called “Slider”.
- Set its minimum to -1.
- Hook up its Out(1) to the Step Node’s In(1).
- If you move the slider bar on the slider node, you will see it affects the step amount in the step node.
- Now turn it into a property (the same way we did the color node) and name it something like “fill amount”.

So now your graph should look like this.



And now you have your first shader. If you make multiple materials with the same shader, and apply them to other meshes, you can get different results.



Challenges:

- Right now, the top color is white, see if you can figure out a way to be able to change that color on the fly the same way we do the fill color.
- Read into the **Time node** and its time sine setting to see how to get the fill to animate up and down
- originally , the cork was brown. See if you can figure out a way to replicate that. It should be pretty similar to how we pulled of the original fill effect (with a **Step Node**), but you will probably need a couple more nodes such as the **multiply Node** and the **invert color node**.

Shadergraph is super powerful, and we've only scratched the surface of what you can do with it (we didn't even touch vertex manipulation). If you're interested, keep experimenting with it, look up other people's graphs on the internet, and
don't be afraid to share what you make.