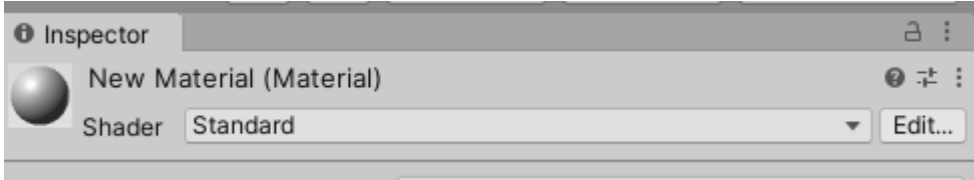


COMP2007 - Game Development

Week 10 - Code session

Shaders

Drawing things to the screen in Unity requires a Material, whether it's a 3D or 2D object or even UI elements. A Material has a Shader selection at the top of the inspector:

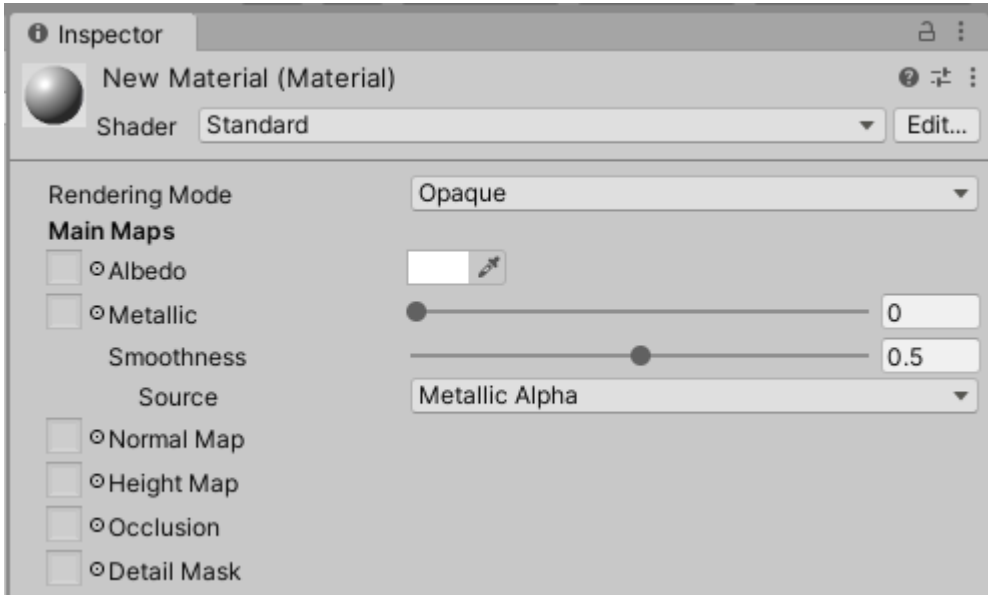


By default, the “Standard” Shader is selected when creating a new Material in Unity.

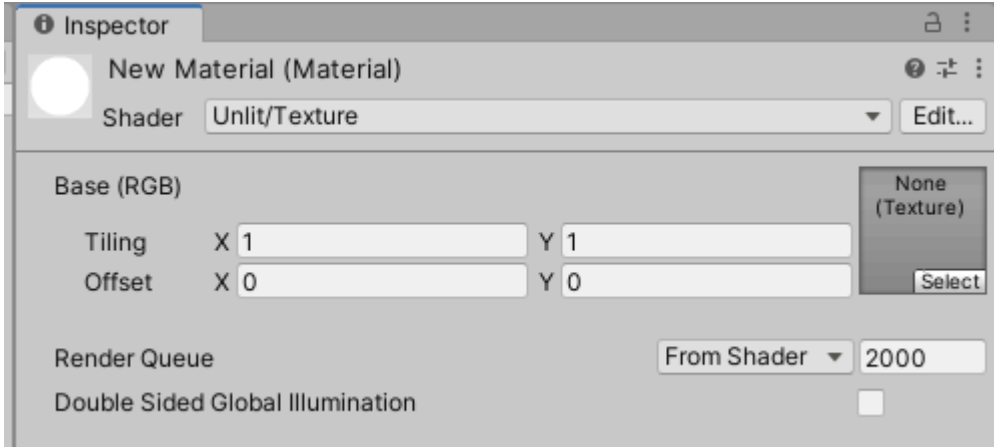
What is a Shader?

A Shader is the algorithm or set of instructions for the material to draw anything to the screen. Similar to a Component, a Shader provides sets of adjustable properties for the material to use, like textures, colours etc. A Shader is a small program, written using a specialised language, HLSL (High-Level Shading Language). The Shader program communicates directly with your computer's CPU and GPU with drawing instructions. Unity uses the HLSL programming syntax to implement its Shaders, but has many custom options specific for use with Unity. Because Unity can export applications to many platforms, there are a huge number of different CPU/GPU setups for its Shaders to consider.

Unity's Standard Shader has a number of settings for many different Material setups



The Unlit Texture Shader only has a few options for setting a texture, its scale (Tiling) and position (Offset)



Shader Graph assets and Code-based Shaders

You can create Shaders in the Shader graph using drag and drop tools, or create a code based shader script. We will focus on the Shader Graph Asset for this tutorial.

Shader Graph Asset

- Requires **Shader Graph package** and **URP** (Universal Render Pipeline) package
- Node-based tool
- No code required to create effects

Code Based Shader

- Uses the **HLSL** (High Level Shader Language) programming language
- Unity Shaders use a custom library built on top of HLSL for effects

Creating a Shader Graph Asset

In the Project view, select Create -> Shader Graph -> URP -> Shader Type

Shader Graph Types

3D Shaders

- Lit
 - Lighting
- Unlit
 - No lighting
- Decal
 - Transparent overlay

2D Shaders

- Sprite Custom Lit
 - Custom lighting
- Sprite Unlit
 - No lighting
- Sprite Lit
 - Lighting applied

Working with Shader Graph Assets

To view the shader on a 3D or 2D assets create a **material** and select the shader from the **Shader dropdown**

In the Project view, a **Shader Graph asset** has an icon like the one below

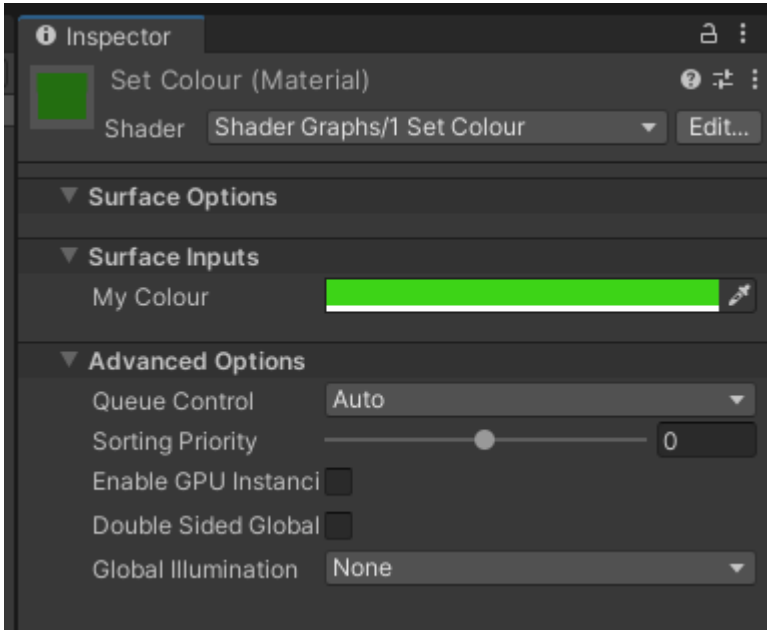


Materials look like this:



When a Shader Graph Asset is selected for the Material, the Inspector should have settings similar to below

NOTE: Your custom settings will appear inside of the Surface and Advanced sections.



The Shader Graph Editor

<https://docs.unity3d.com/Packages/com.unity.shadergraph@12.1/manual/Create-Shader-Graph.html>

1 Set Colour

Shader Graphs

+

My Colour

Color

Variable node

My Colour(4)

Properties

X 0

X 0.5

HDR

X 1

X 1

X 0.5

Vertex

Object Space

Position(3)

Object Space

Normal(3)

Object Space

Tangent(3)

Fragment

Base Color(3)

Tangent Space

Normal (Tangent Space)(3)

Metallic(1)

Smoothness(1)

Emission(3)

Ambient Occlusion(1)

Alpha(1)

Alpha Clip Threshold(1)

Graph Inspector

Node Settings

Graph Settings

Precision

Single


Target Settings

Active Targets

Universal

Universal

Main Preview



Blackboard

Contains Variables

Inspector

Shows graph and node settings

Preview

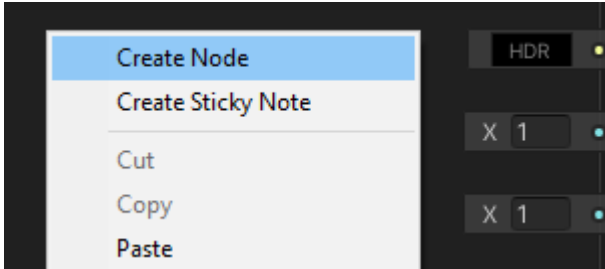
Shows Shader output

Master Stack

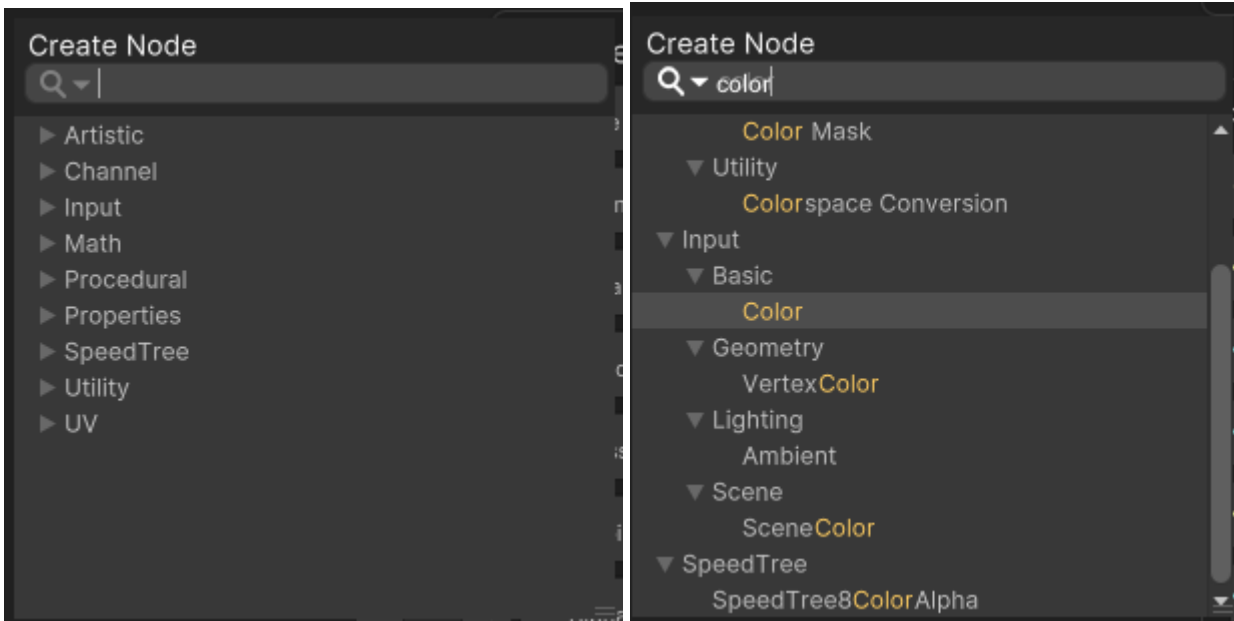
Endpoint nodes for all Shader properties

Adding nodes

Right click in the Graph view and select **Create Node**



Use the search bar for a specific node or select from the drop down list



About Nodes

Inputs and Outputs

Nodes can accept data as a form of input which will often be a primitive type, such as a float, int or common structures like Vector3, Color etc. Output is often the same data type with adjusted values, such as a change to a float over time or a change in colour.

Think of nodes as C# methods, they accept some form of input parameters, will do some processing and return some result data. Nodes can be chained together to make many adjustments to data for a variety of results.

Data types for inputs and outputs are indicated by **colour**

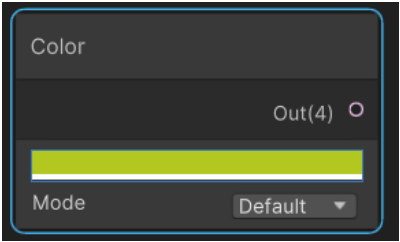
- Float = blue
- Boolean = purple
- Vector2 = green
- Vector3 = tan
- Color = pink

The amount of primitive data types is displayed in brackets beside the input or output
For example Color has 4 floats (red,green,blue and alpha)

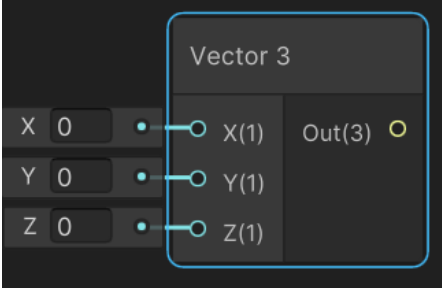
Inputs Each input value below is a single float	Outputs Float is 1 float value Vector3 is 3 float values Color is 4 float values

Example nodes

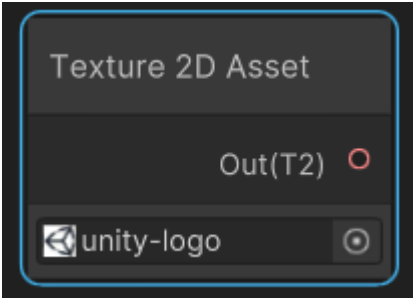
The **Color node** has a Colour picker at the bottom and outputs to the right.
NOTE: you can customise the colour using the picker



The Vector3 node has 3 float inputs on the left and a single vector3 output on the right

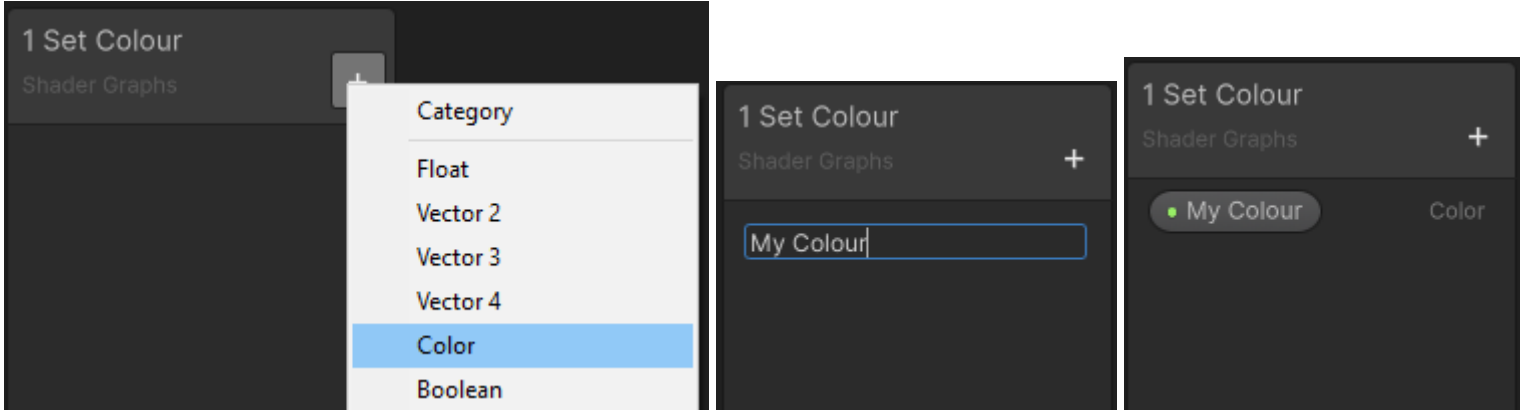


The **Texture2D Asset node** uses an image from your project (inserted in the inlet at the bottom of the node) and will output that image as **Texture2D data**.

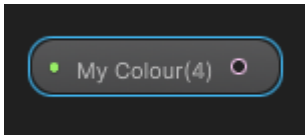


Blackboard properties

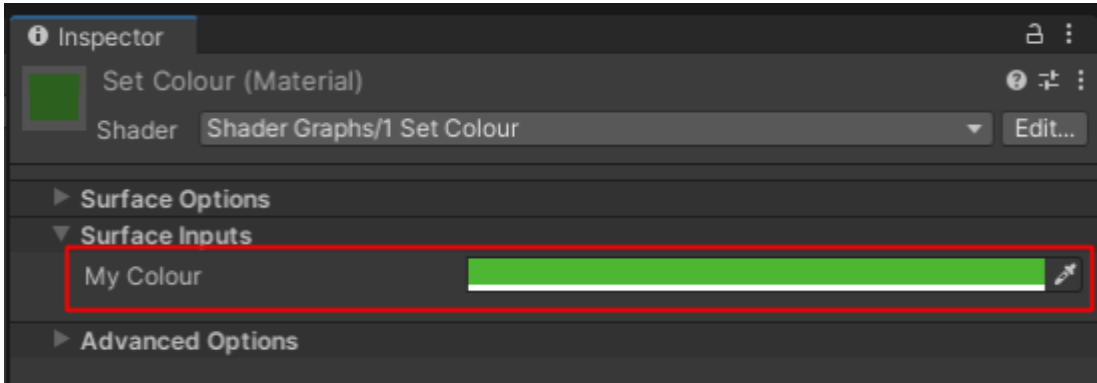
You can create custom properties for use in any material that uses the shader
Create a property by pressing the + button at the top of the blackboard window.
Enter a name for your property and press enter



To use your property in the graph, drag it from the blackboard to the graph.
It will look like the image below, but the outputs will be different according to the property type.



Blackboard Properties are “public”, we can edit them on any material using our shader in the Unity Inspector and set the property using C# code



To connect the property to another node, drag from the output to the desired input

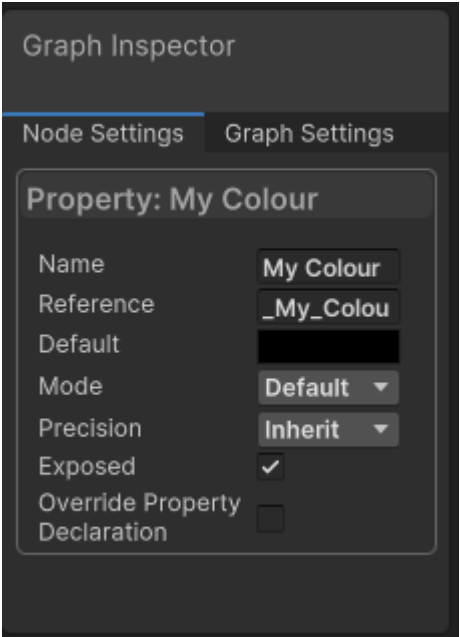


NOTE: the pink coloured output of the property is different to the tan coloured input.
The input is a vector 3 with 3 floats, so the 4th float of the colour will be dropped
The conversion from colour to vector3 is as follows:
Red = X
Green = Y
Blue = Z
Alpha = dropped

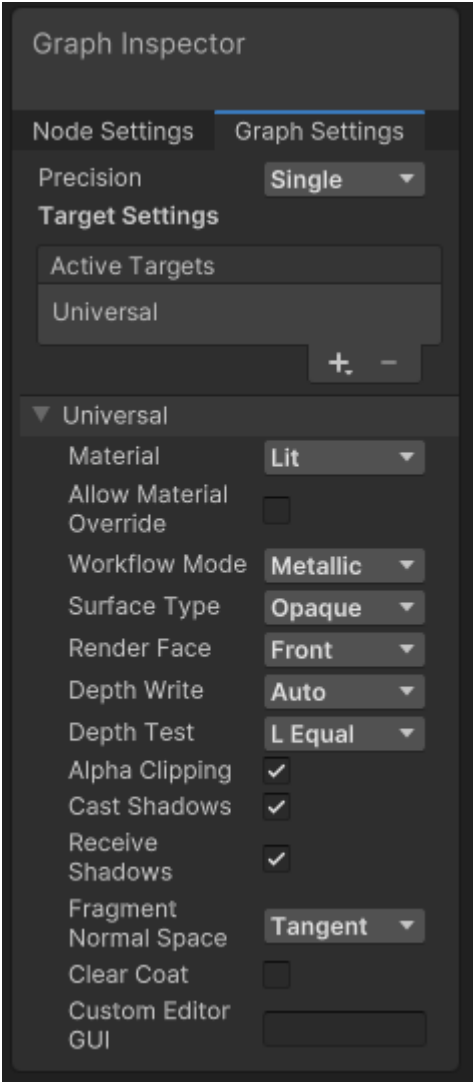
So our alpha value for the colour is dropped by the input for the, input.

Graph Inspector

Inspect nodes and custom properties. You can provide default settings for your blackboard properties here.



Graph settings allow you to change the type of shader if required.
For example if I wanted to change to an Unlit shader, I can set it from the **Material** dropdown below



References

URP (Universal Render Pipeline) Documentation
<https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@12.1/manual/index.html>

Shader Graph Documentation
<https://docs.unity3d.com/Packages/com.unity.shadergraph@12.1/manual/index.html>

URP Shader Types
Lit Shader
<https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@12.1/manual/lit-shader.html>

Unlit
<https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@12.1/manual/unlit-shader.html>

Sprite Shaders (2D)
<https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@12.1/manual/ShaderGraph.html>

Decal
<https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@12.1/manual/decal-shader.html>

About code based shaders
Writing Code-based Shaders in Unity
<https://docs.unity3d.com/2021.2/Documentation/Manual/SL-ShadingLanguage.html>

HLSL (High Level Shader Language)
<https://docs.microsoft.com/en-us/windows/win32/direct3dhls/dx-graphics-hlsl>

Freya Holmer in-depth Unity shader tutorials
<https://www.youtube.com/c/Acegikmo>

Shadertoy - Shader community and creation tool
<https://www.shadertoy.com/>