Game coding tutorials

1. Third person character controller

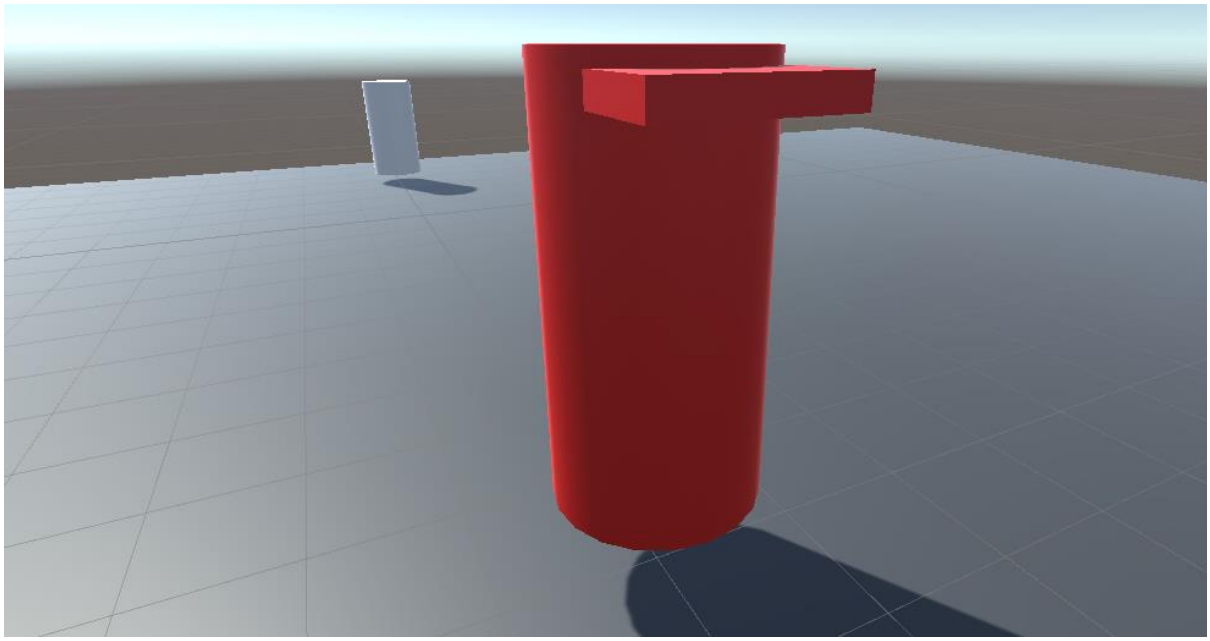
2. Speed Boost

3. Cel Shader

4. Main Menu

Third Person Character Controller

1. Set up an empty object for the player in a game scene, like below and lace the 3d objects you want to make the model I them inside of the empty game object.



2. Put a rigid body on the empty game object along with a character controller.

3. After that We'll want to make a third person movement Script this can be done by using the following code

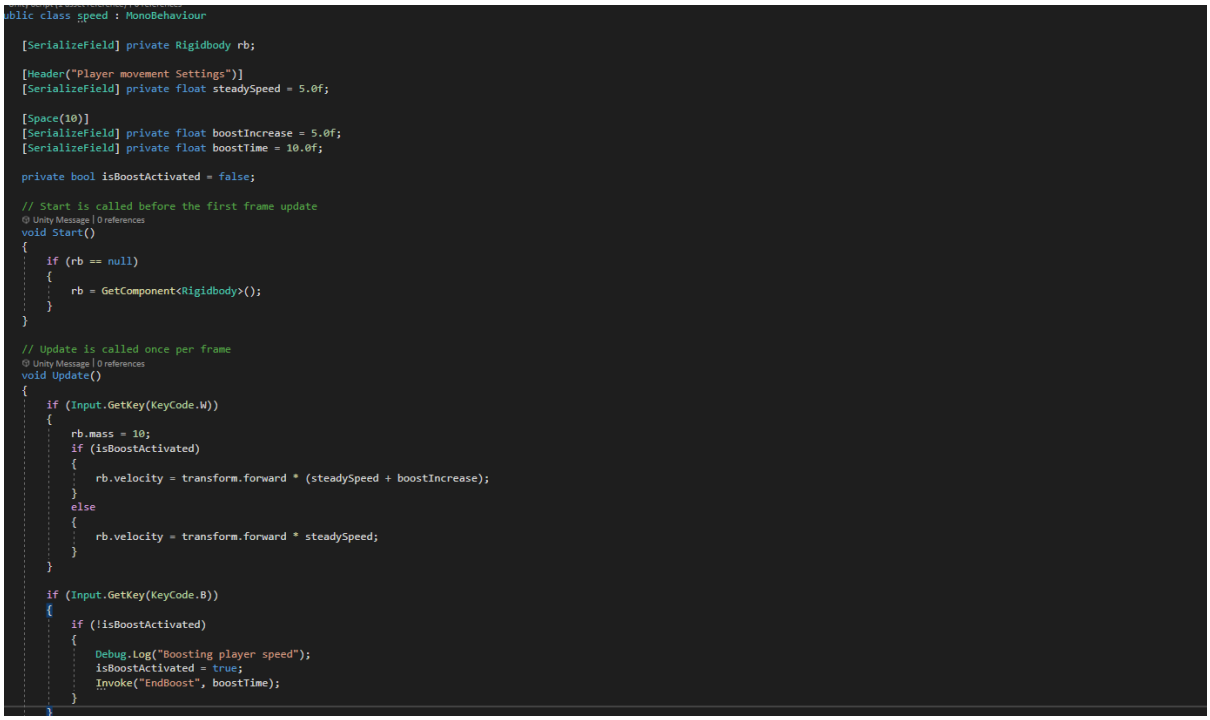
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity Script (1 asset reference) | 0 references
public class thridPersonMovement : MonoBehaviour
{
    public CharacterController controller;
    public Transform cam;
    public float speed = 6f;

    public float turnsmoothtime = 0.1f;
    float turnsmoothvelcoity;

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        float horizontal = Input.GetAxisRaw("Horizontal");
        float vertical = Input.GetAxisRaw("Vertical");
        Vector3 direction = new Vector3(horizontal, 0f, vertical).normalized;

        if (direction.magnitude >= 0.1f)
        {
            float targerAngle = Mathf.Atan2(direction.x, direction.z) * Mathf.Rad2Deg + cam.eulerAngles.y;
            float angle = Mathf.SmoothDampAngle(transform.eulerAngles.y, targerAngle, ref turnsmoothvelcoity, turnsmoothtime);
            transform.rotation = Quaternion.Euler(0f, angle, 0f);

            Vector3 moveDir = Quaternion.Euler(0f, targerAngle, 0f) * Vector3.forward;
            controller.Move(moveDir.normalized * speed * Time.deltaTime);
        }
    }
}
```

4. The reason we set up a public float for speed is so we can adjust our speed in the inspector. We set up the public transform cam so we can drag our camera in so it can follow us. We then set two floats for horizontal and vertical that equal input gets raw axis. What this function does is it looks on the keyboard for the input keys and it set the values as -1 or 1 and if they are those values if moves the player, the reason it is on void update as it is being called every frame.

   Under this we have an if statement the purpose of the is to make the character turn smoother the reason we have if magnitude is >= 1 is we only want it to place when we are turning.

Speed Boost

To set this up we must create a new script.

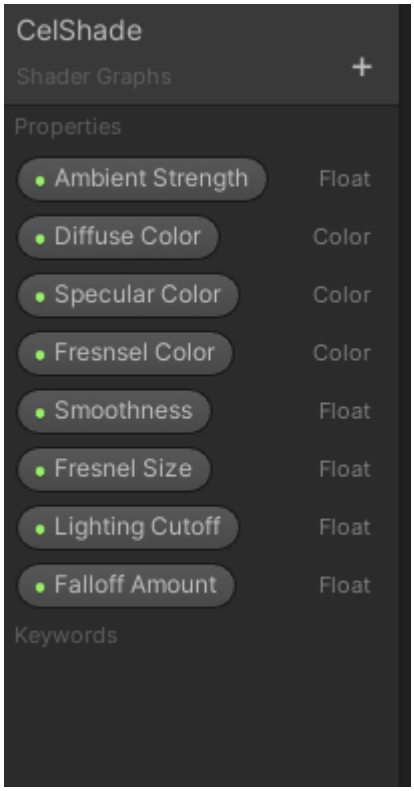


```
ublic class speed : MonoBehaviour

    [SerializeField] private Rigidbody rb;

    [Header("Player movement Settings")]
    [SerializeField] private float steadySpeed = 5.0f;

    [Space(10)]
    [SerializeField] private float boostIncrease = 5.0f;
    [SerializeField] private float boostTime = 10.0f;

    private bool isBoostActivated = false;

    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {
        if (rb == null)
        {
            rb = GetComponent<Rigidbody>();
        }
    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        if (Input.GetKey(KeyCode.W))
        {
            rb.mass = 10;
            if (isBoostActivated)
            {
                rb.velocity = transform.forward * (steadySpeed + boostIncrease);
            }
            else
            {
                rb.velocity = transform.forward * steadySpeed;
            }
        }

        if (Input.GetKey(KeyCode.B))
        {
            if (!isBoostActivated)
            {
                Debug.Log("Boosting player speed");
                isBoostActivated = true;
                Invoke("EndBoost", boostTime);
            }
        }
```

For this script we set a header with a name of player movement settings the reason for this we want a set of different floats to be able to change in the inspector and this is a nice way of grouping them. Then set up a another with a rigid body and finally a private bool of isboostActived active and set it to false. Is void start you want to set our rigid body to null and get component of rigid body.

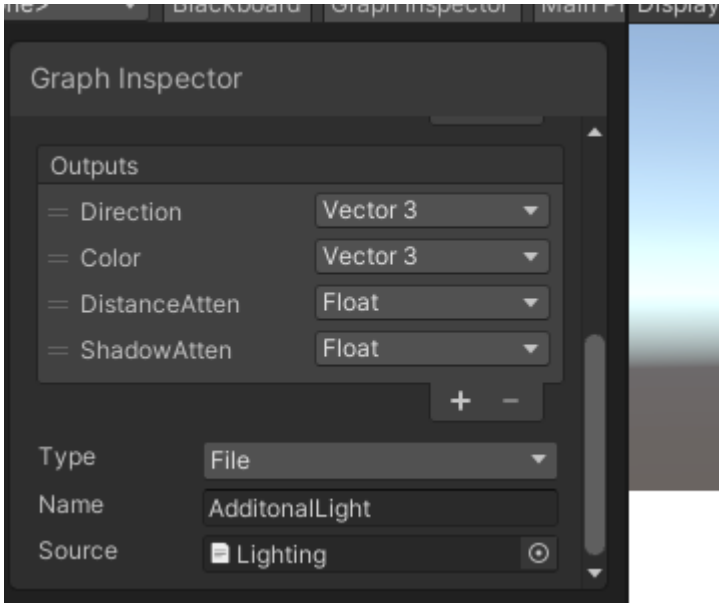
Then in the void update we want to use a input keycode on w and check if boost is active and the well use a velocity = to transform froward to give use monument. And a another to active the speed on gets keycode b to active the speed boost. So it works when using w acts move forward like most games would and then when you press and let go of b you boost

Cel shader



So, the first step for the cel shader is to create the following Variables in the Shader Graph

But before this we'll need to make a custom .hlsl file for out custom functions, these will be dragged into the graph inspector when clicked on the additional function. It should look this below; this will be done for our second custom function as well



After this we'll need to program the .hlsl file(lighting in mine)

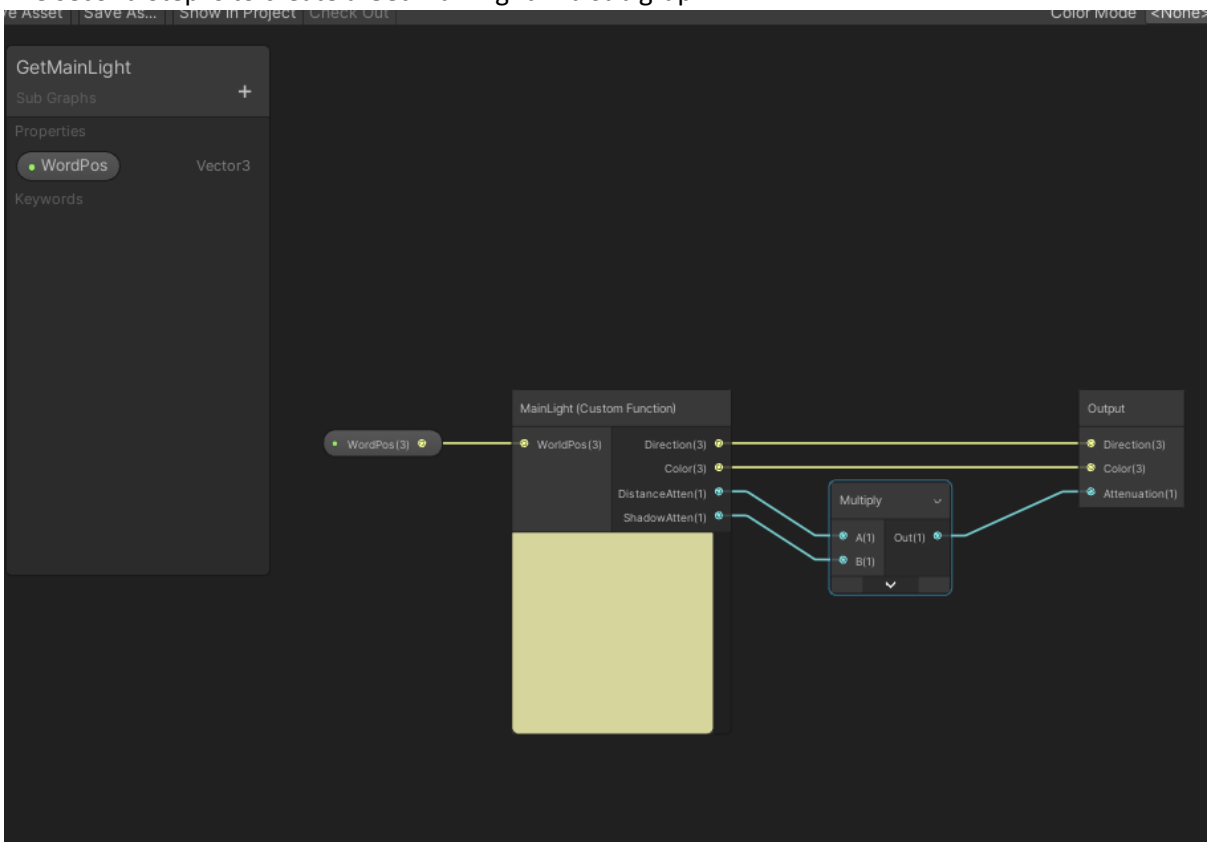
```
void MainLight_float(float3 WorldPos, out float3 Direction, out float3 Color,
    out float DistanceAtten, out float ShadowAtten)
{
#ifdef SHADERGRAPH_PREVIEW
    Direction = normalize(float3(0.5f, 0.5f, 0.25f));
    Color = float3(1.0f, 1.0f, 1.0f);
    DistanceAtten = 1.0f;
    ShadowAtten = 1.0f;
#else
    float4 shadowCoord = TransformWorldToShadowCoord(WorldPos);
    Light mainLight = GetMainLight(shadowCoord);

    Direction = mainLight.direction;
    Color = mainLight.color;
    DistanceAtten = mainLight.distanceAttenuation;
    ShadowAtten = mainLight.shadowAttenuation;
#endif
}

void MainLight_half(half3 WorldPos, out half3 Direction, out half3 Color,
    out half DistanceAtten, out half ShadowAtten)
{
#ifdef SHADERGRAPH_PREVIEW
    Direction = normalize(half3(0.5f, 0.5f, 0.25f));
    Color = half3(1.0f, 1.0f, 1.0f);
    DistanceAtten = 1.0f;
    ShadowAtten = 1.0f;
#else
    half4 shadowCoord = TransformWorldToShadowCoord(WorldPos);
    Light mainLight = GetMainLight(shadowCoord);

    Direction = mainLight.direction;
    Color = mainLight.color;
    DistanceAtten = mainLight.distanceAttenuation;
    ShadowAtten = mainLight.shadowAttenuation;
#endif
}
```

For this scetion of the code it’s very impotant to copy it extactly or many of the lighting on a maetrial won’t work. A lot of this is setting different attirbutes to what we want to get it behave in a way that a cel shader does.

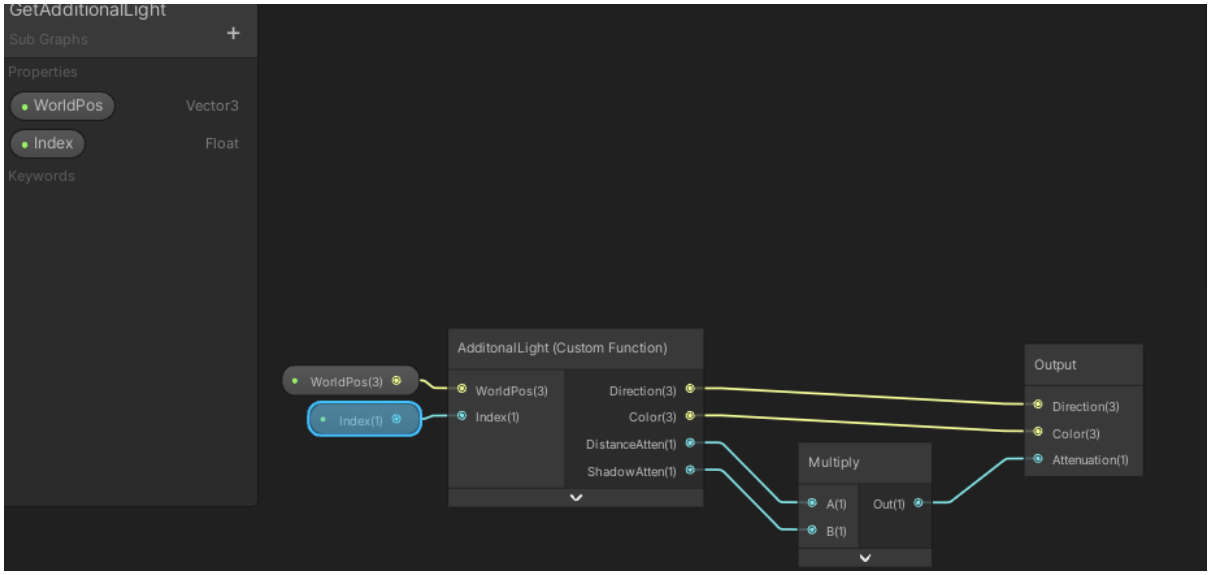
The seocnf half of the code.

```
    ShadowAtten = mainLight.shadowAttenuation;
#endif
}

void MainLight_half(half3 WorldPos, out half3 Direction, out half3 Color,
    out half DistanceAtten, out half ShadowAtten)
{
#ifdef SHADERGRAPH_PREVIEW
    Direction = normalize(half3(0.5f, 0.5f, 0.25f));
    Color = half3(1.0f, 1.0f, 1.0f);
    DistanceAtten = 1.0f;
    ShadowAtten = 1.0f;
#else
    half4 shadowCoord = TransformWorldToShadowCoord(WorldPos);
    Light mainLight = GetMainLight(shadowCoord);

    Direction = mainLight.direction;
    Color = mainLight.color;
    DistanceAtten = mainLight.distanceAttenuation;
    ShadowAtten = mainLight.shadowAttenuation;
#endif
}

void AdditionalLight_float(float3 WorldPos, int Index, out float3 Direction,
    out float3 Color, out float DistanceAtten, out float ShadowAtten)
{
    Direction = normalize(float3(0.5f, 0.5f, 0.25f));
    Color = float3(0.0f, 0.0f, 0.0f);
    DistanceAtten = 0.0f;
    ShadowAtten = 0.0f;

#ifndef SHADERGRAPH_PREVIEW
    int pixelLightCount = GetAdditionalLightsCount();
    if(Index < pixelLightCount)
    {
        Light light = GetAdditionalLight(Index, WorldPos);

        Direction = light.direction;
        Color = light.color;
        DistanceAtten = light.distanceAttenuation;
```

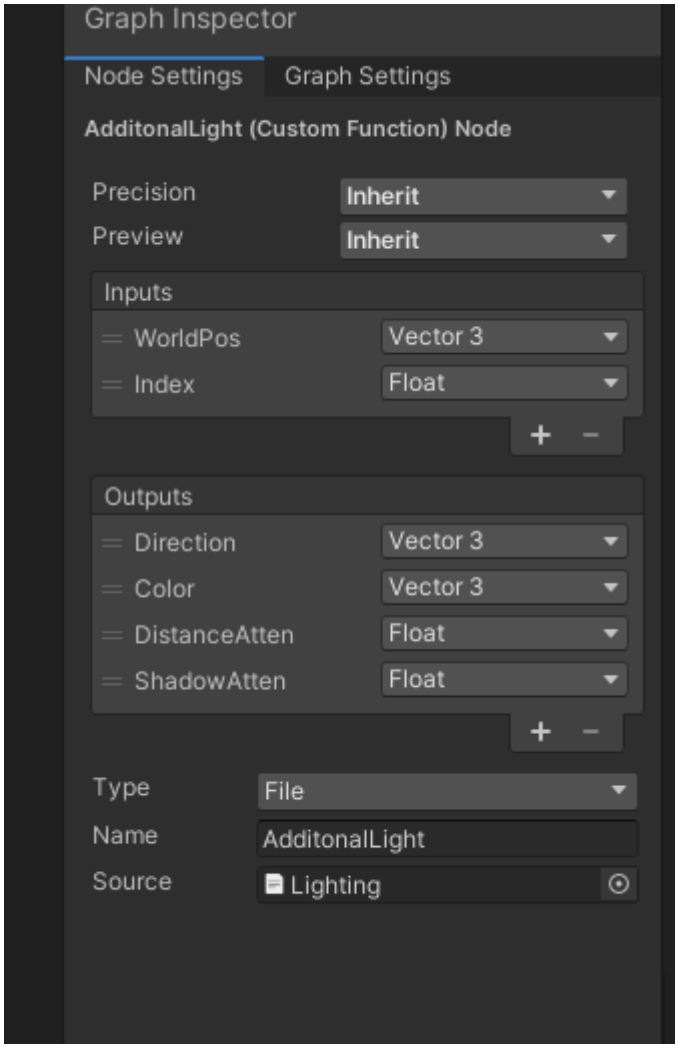
The second Step is to create a Get main light in a sub graph



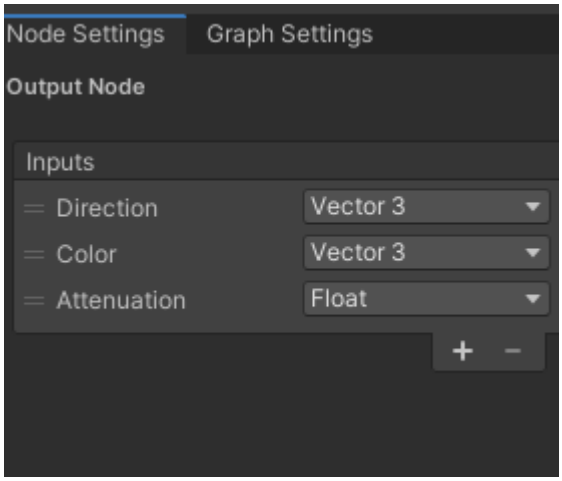
After this you want go to the main graph again and right click create node then create sub node then get additional light.



The first step is dragging the two variables we made into the graph. Then the step after this is to make a custom function and we'll call it Additional light. Then Click on the custom function, then the graph inspector and add and set the following input and outputs.



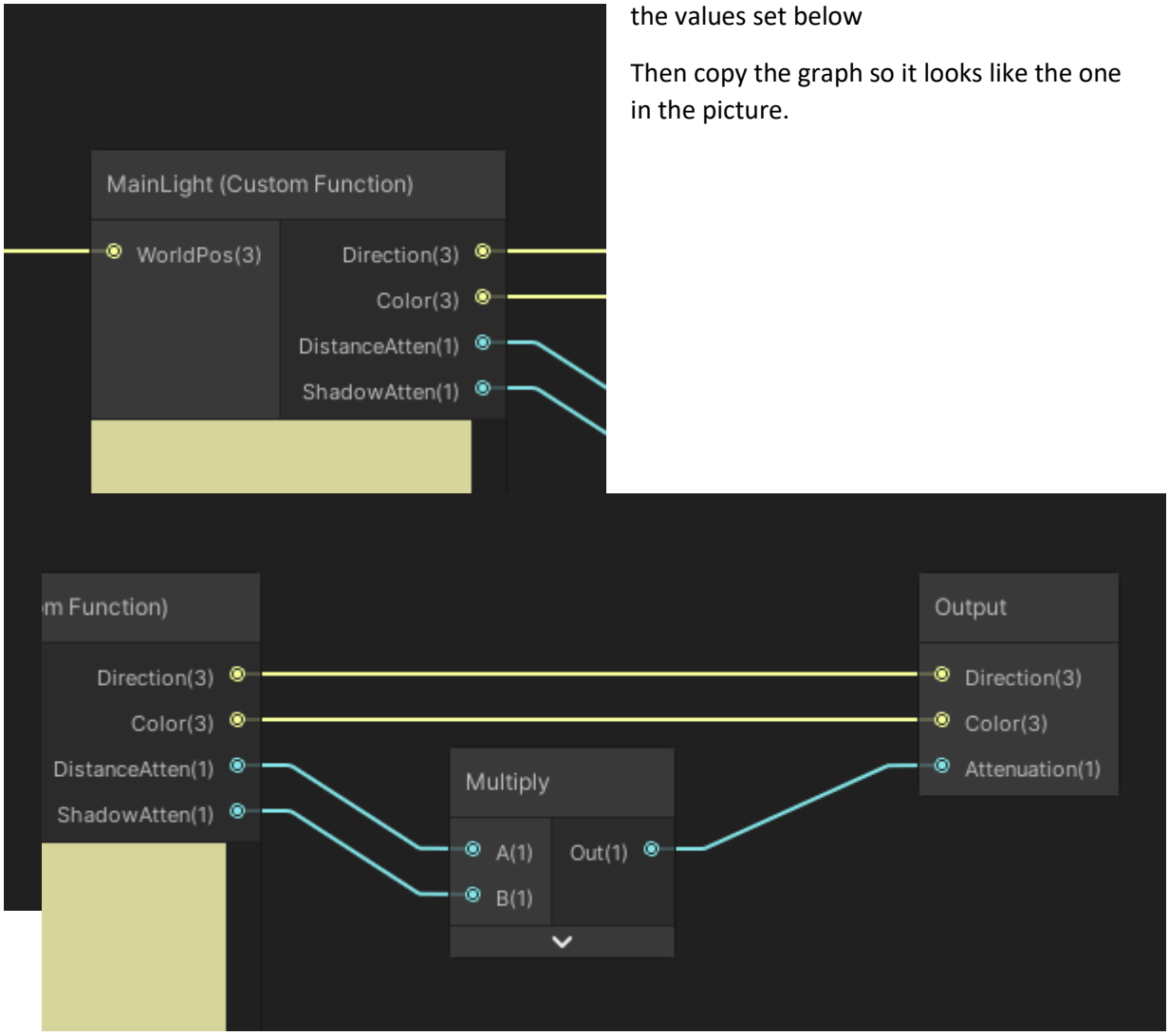
Then you want to make a multiply node and connect DistanceAten and ShadowAtten to it, then create a output node and make the following outputs.



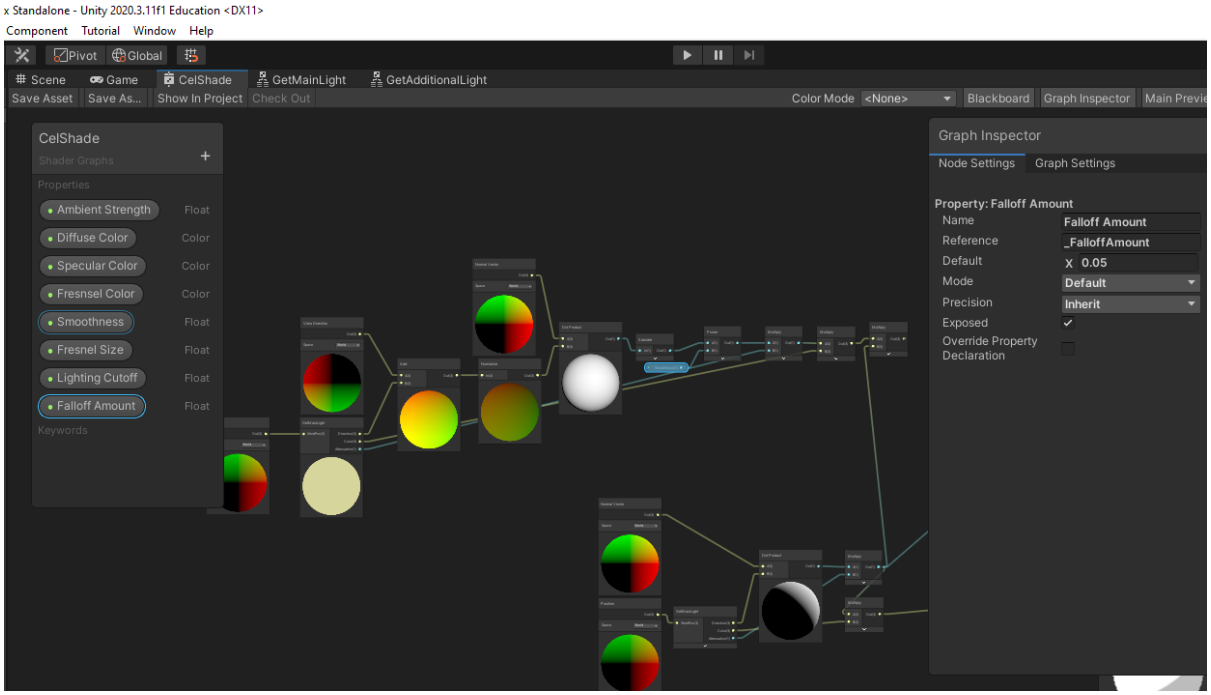
Set the following in the output node.

Then go the main graph and create another sub graph we'll call this getMainLight and we'll create a property of wordPos, then we'll create a custom function and fill the outputs we'll set the outputs to the values set below

Then copy the graph so it looks like the one in the picture.



We'll want to go to the main shader graph and add these values



The values we want to set:

Ambient strength 0.5

Diffuse colour: set the colour to white In the inspector

Specular colour: set the colour to black In the inspector

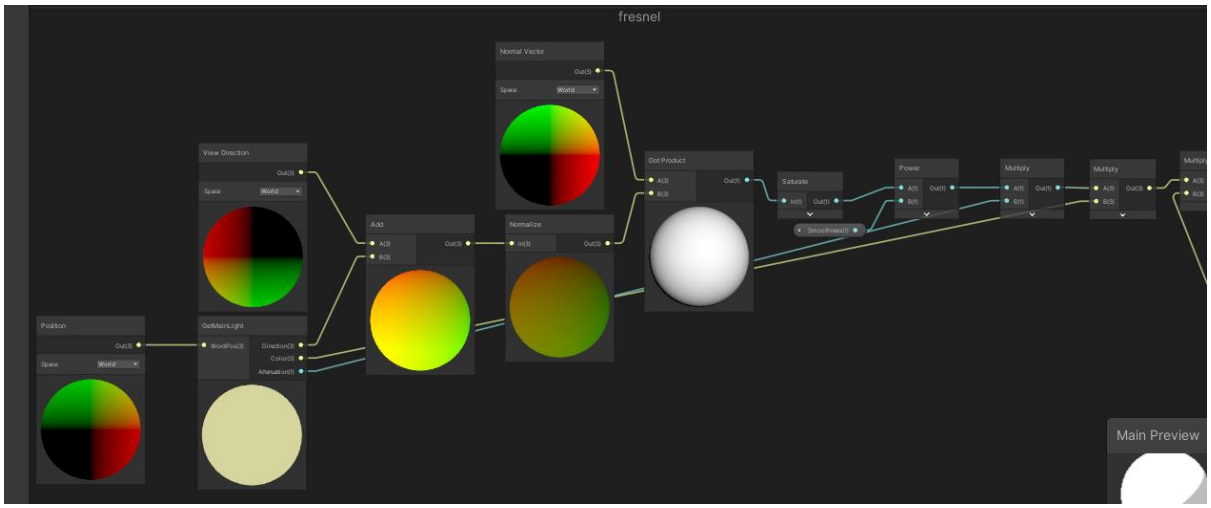
Fresnsel Colour: set to white

Smoothness: leave as 0

Fresnel size: 0.1

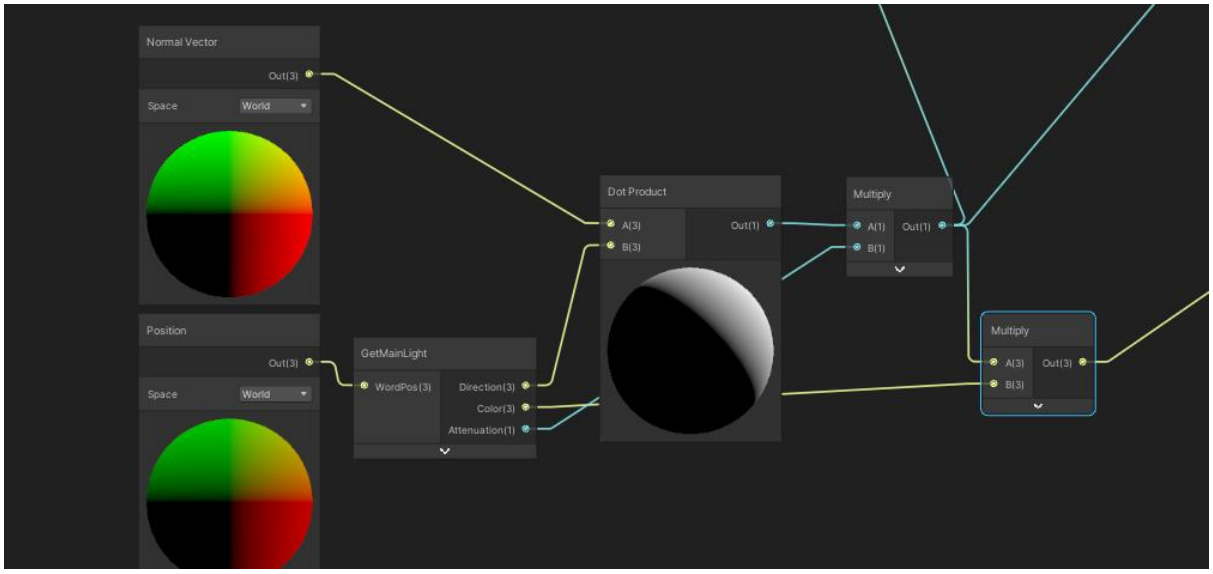
Lighting cutoff: leave as 0

Fallof Amount: 0.05

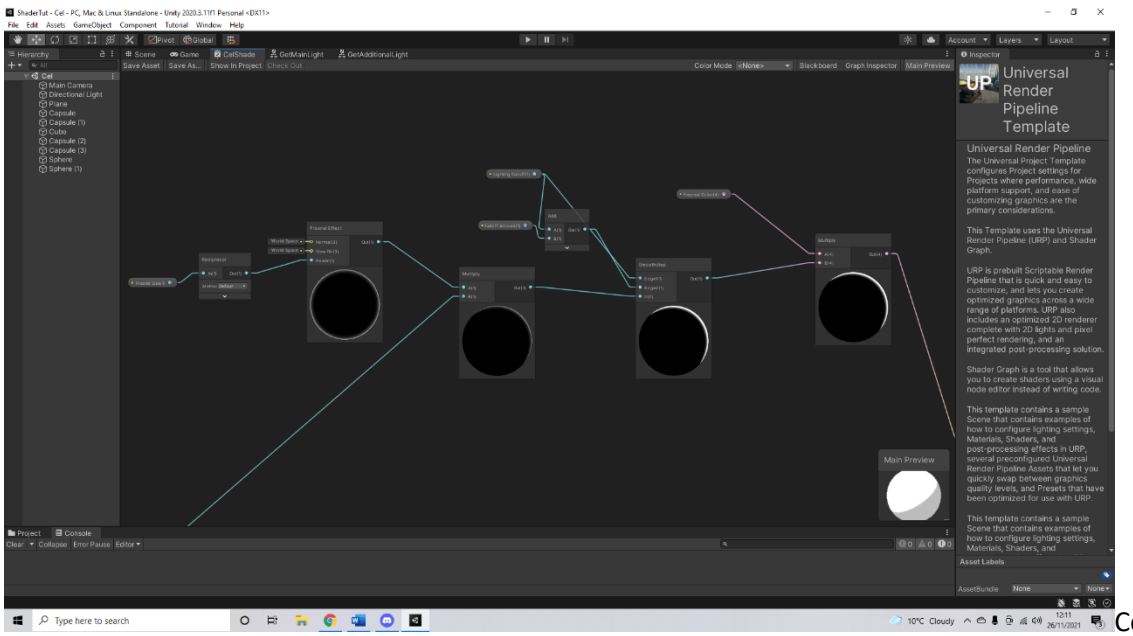


Then you'll want to copy the part of the graph below, so it matches the imagine above.

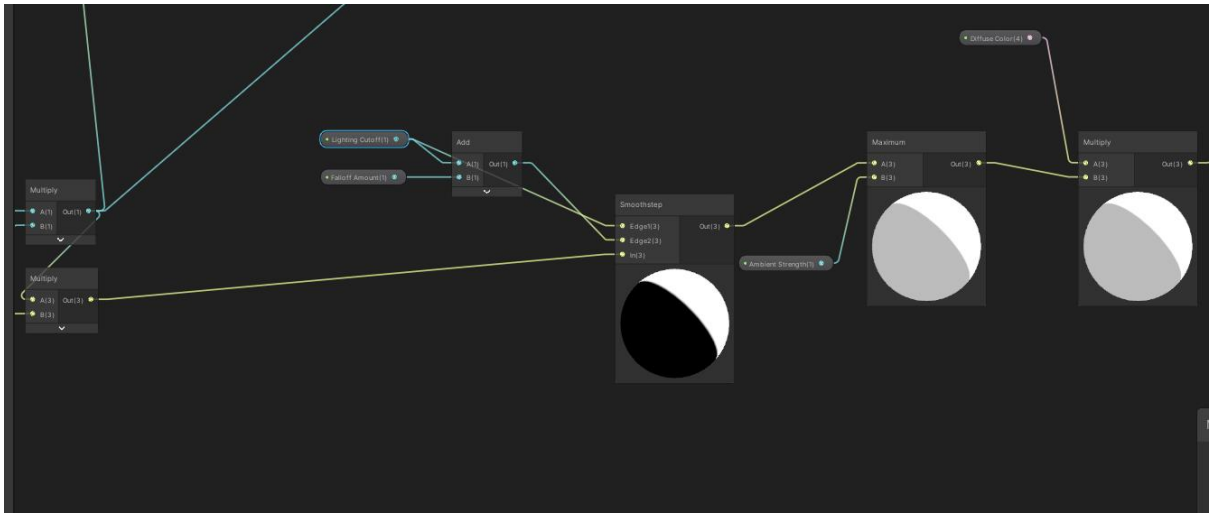
The next step is to copy this section



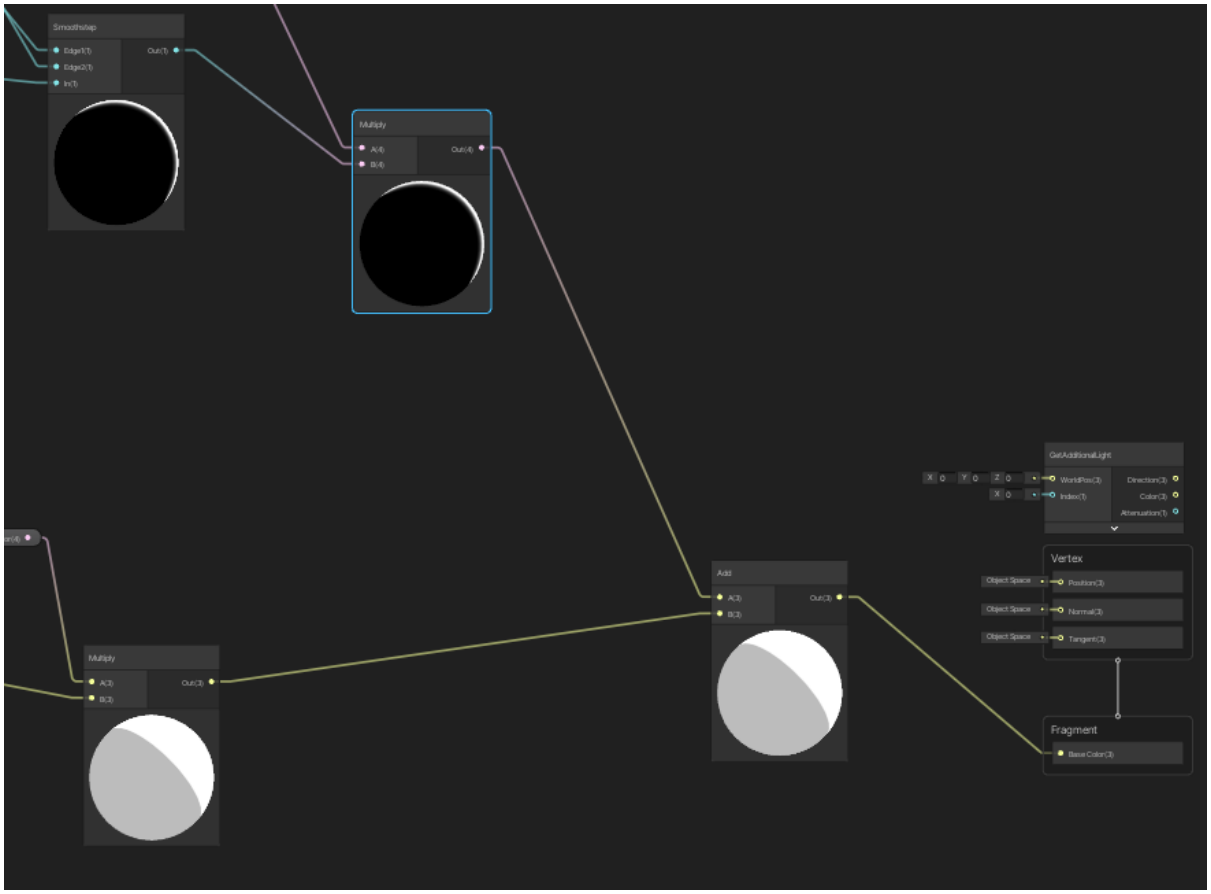
We'll want to drag the multiply b out multiply out that comes from dot product and then another line of it then that will go to the other yellow multiply in a slot b.



Copy the values of the shader graph into your own. The Blue line comes the previous imagine of the multiply that trails off in a blue line.

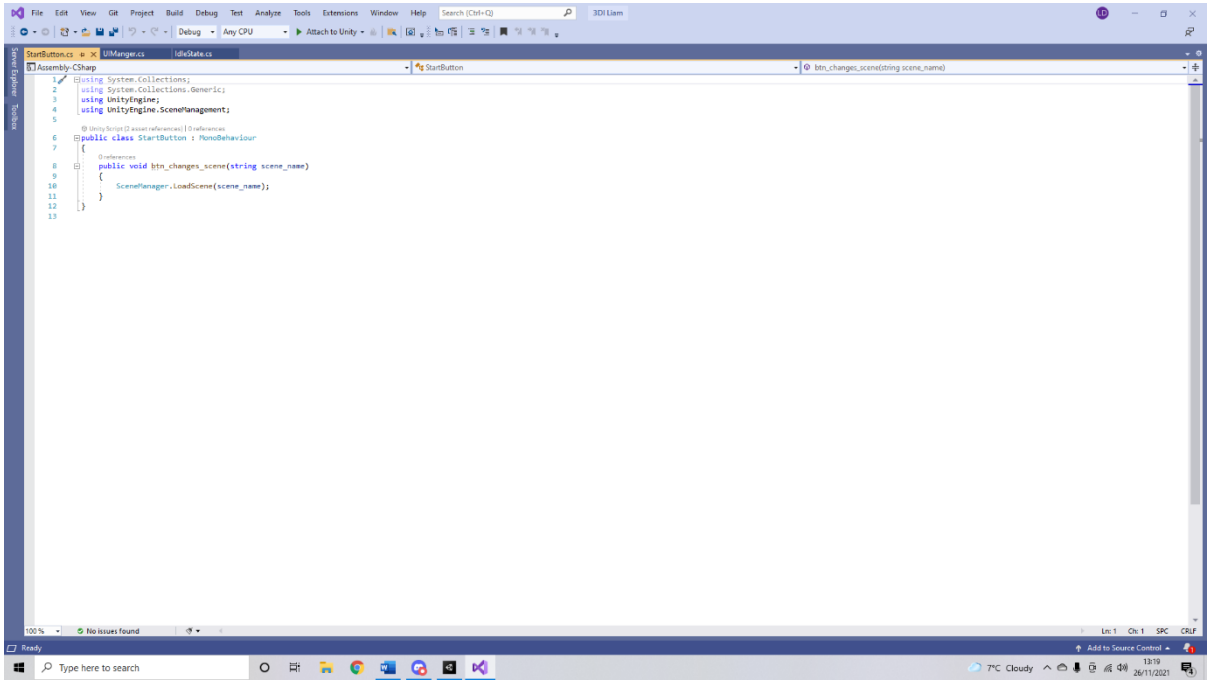


Once again, we'll want to copy the values into the shader graph again with the nodes in the picture above.

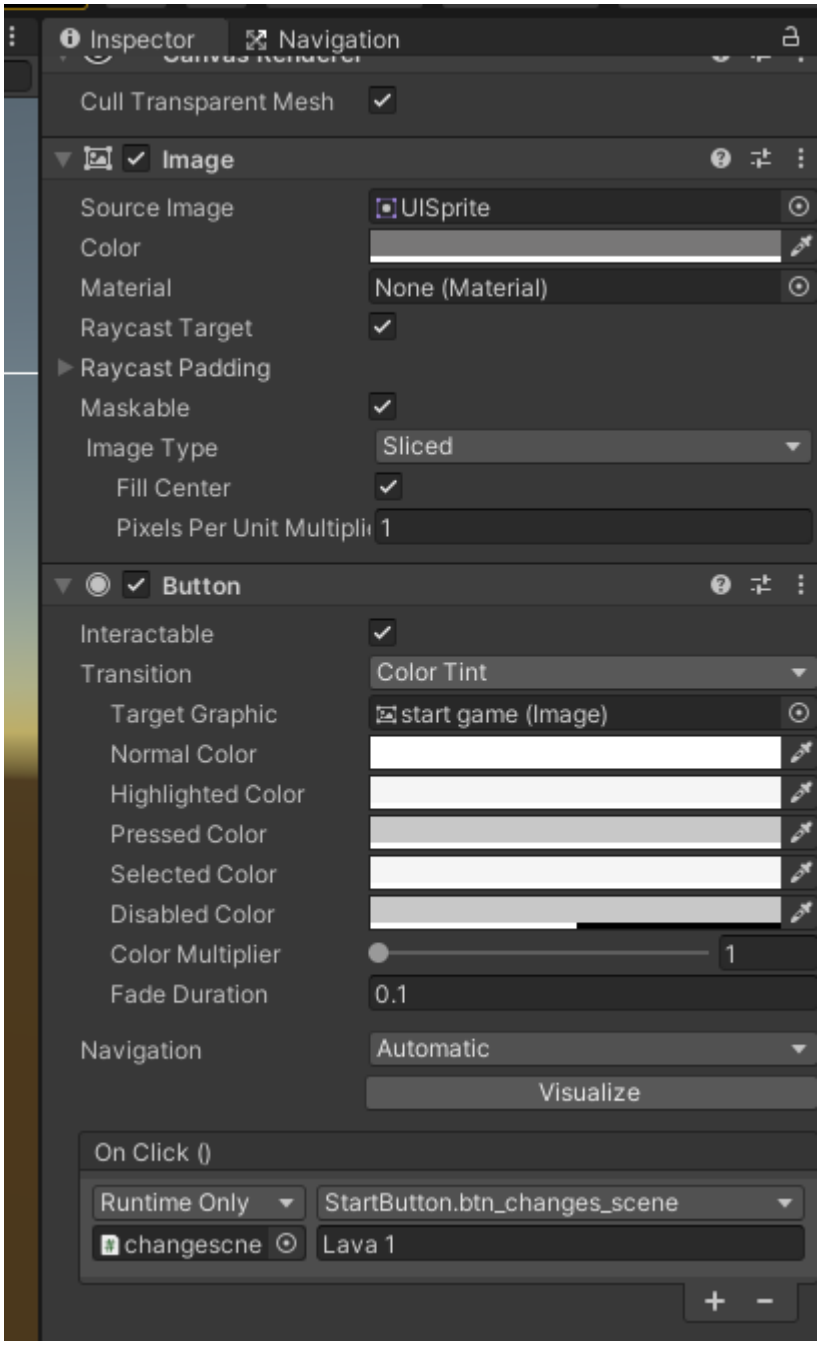


Now for the last time put these values in the shader graph and it should be done.

Main Menu



The code needed for this relatively simple and this all we need so copy this out.



The next thing to do is create a a ui element with a button then attach the script previously made onto an empty game object drag to an on click element and set to run time and with the function seen above.