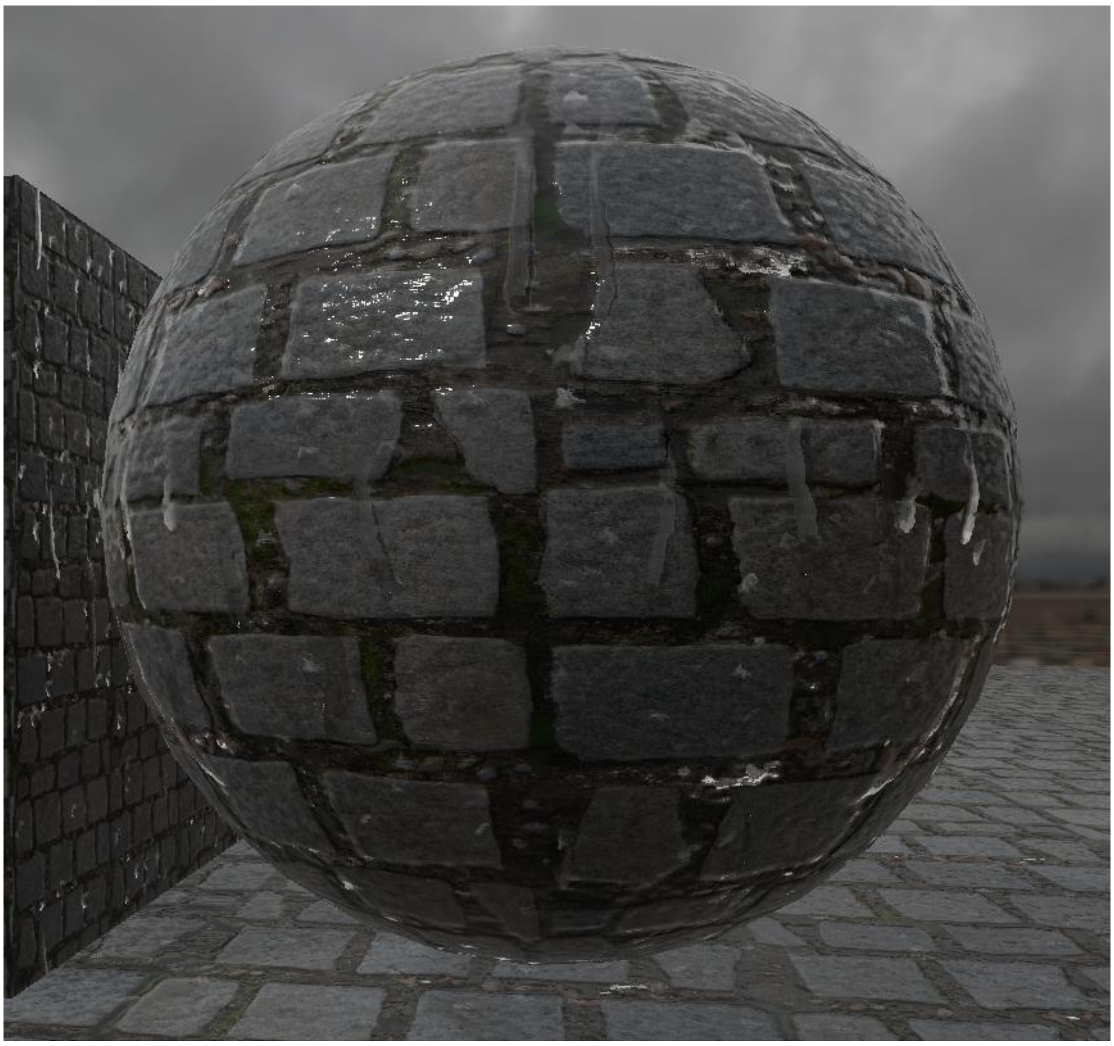
Raindrop Materials For URP

1.Abstract

This package contains a material of the rain drop effect,which includes the effect of water drop sliding off the surface,and the effect of randomly generated water spots.It also included an editor tool which can help generating custom rain drop textures.



2.Directory Introduction

All the assets of this package were placed in the ‘Assets/NekoPunch/Raindrop Materials For URP’ directory. This directory includes several sub directories:

Documents: Stores this document.

Materials: Stores all the materials of this package. BakeDropTex.mat is used for baking the raindrop texture. While RainDrop.mat and RainDrop_Floor.mat is the example materials used to show the object with raindrops on their surface. And Skybox.mat is a skybox material as it’s named.

Meshes: Stores a raindrop mesh, RainDrop.fbx. We will use this mesh to render a mask of raindrops.

Prefabs: Stores a prefab of a GameObject, with CaptureNormal script attached.Used for generating the raindrop mask texture.

Scenes: Stores a sample scene, SampleScene.unity

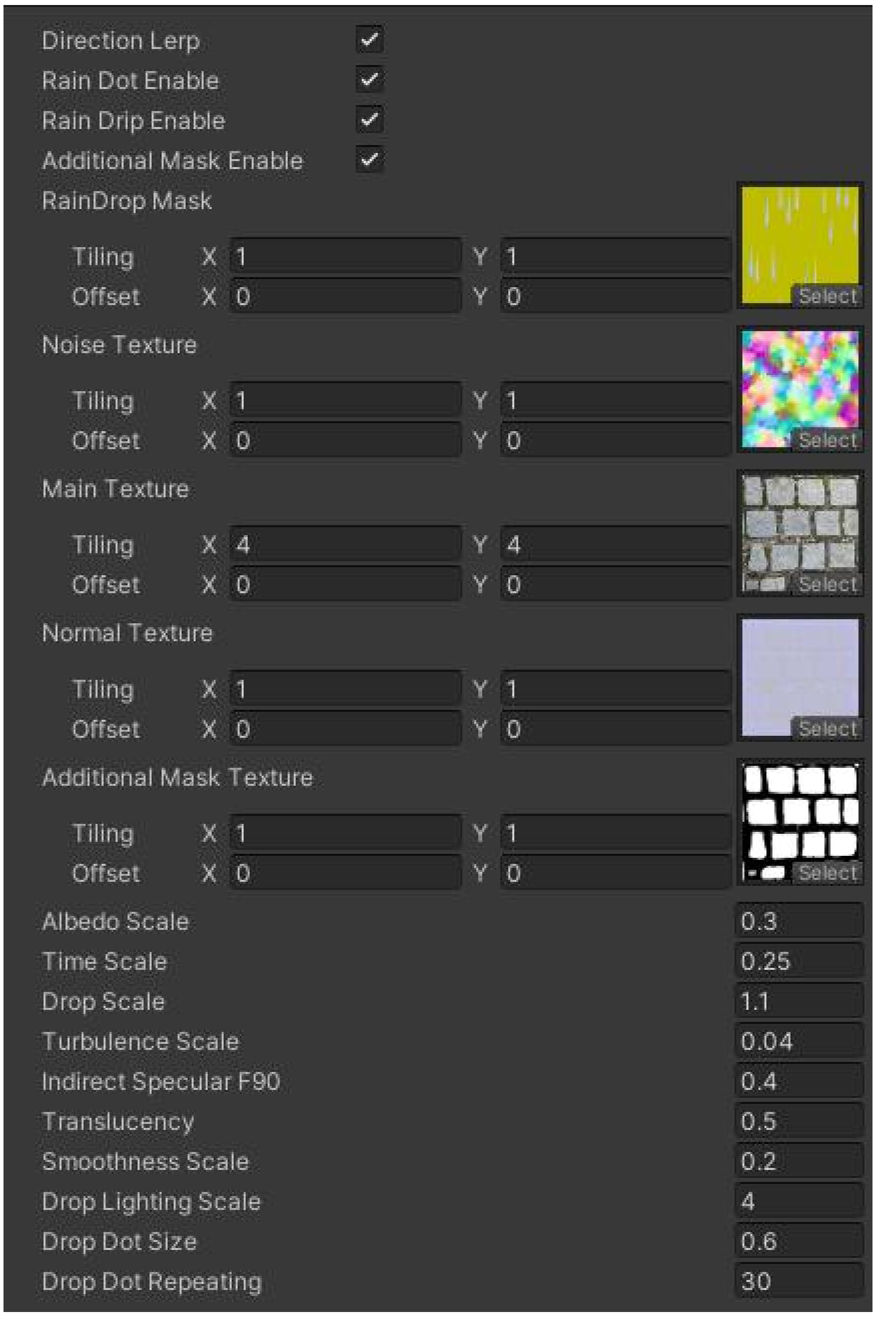
Scripts: Stores 2 c# scripts: 1. CaptureNormal.cs, used for generating the raindrops mask texture.2.RaindropsObject.cs,used to manage some shader parameters for the Direction Lerp Mode.

Shaders: Stores all the shaders of this package. DropNormalOutput.shader is the shader used for rendering normal and mask of raindrops. Raindrop.shader is the shader of a standard lighting material with raindrop effect added. And RainDrop.hlsl is the main library of raindrop effects, you can include this file to implement your own shaders.

Textures: Stores all the textures of this package. CurlNoise.tga is a noise texure used for distorting the move direction of raindrops. output.png is the raindrops' normal and mask texture. You can generate your own version of this texture by using CaptureNormal.cs. The rest of textures are the textures for the demo scene.

3. Quick Start

You can quickly get the raindrop effect by setting RainDrop.mat to the object you want it to get wet on. Here is the material inspector:



Direction Lerp	✓				
Rain Dot Enable	✓				
Rain Drip Enable	✓				
Additional Mask Enable	✓				
RainDrop Mask					
Tiling	X	1	Y	1	
Offset	X	0	Y	0	Select
Noise Texture					
Tiling	X	1	Y	1	
Offset	X	0	Y	0	Select
Main Texture					
Tiling	X	4	Y	4	
Offset	X	0	Y	0	Select
Normal Texture					
Tiling	X	1	Y	1	
Offset	X	0	Y	0	Select
Additional Mask Texture					
Tiling	X	1	Y	1	
Offset	X	0	Y	0	Select
Albedo Scale					0.3
Time Scale					0.25
Drop Scale					1.1
Turbulence Scale					0.04
Indirect Specular F90					0.4
Translucency					0.5
Smoothness Scale					0.2
Drop Lighting Scale					4
Drop Dot Size					0.6
Drop Dot Repeating					30

Direction Lerp: Update and lerp the rain drops from to directions according to the time interval.It is useful when you are dealing with dynamic objects.When you open this toggle,you should attach the RaindropsObject MonoBehaviour to your target GameObject to make it work correctly.

Rain Dot Enable: Enable the rain dot effect or not.

Rain Drip Enable: Enable the rain drop sliding effect or not.

Additional Mask Enable: Enable the additional mask texture or not. If additional mask is enabled, and additional mask texture is used to mask out rain drop areas, rain drops will not appear in areas where the mask pixels are black.

RainDrop Mask: RainDrop Mask Texture,can be generated by CaptureNormal.cs.

Noise Texture: Turbulence noise texture,used for distorting raindrops' moving direction.

Main Texture: Albedo texture of the standard material.

Normal Texture: Normal map of the standard material.

Additional Mask Texture: The additional mask texture used when the additional mask is enabled.

Albedo Scale: Albedo intensity scale

Time Scale: The scale of the raindrops' sliding down speed

Drop Scale: The scale of raindrops' size

Turbulence Scale: The scale of turbulence texture, can control the frequency of raindrops' distortion

Indirect Specular F90: This parameter controls the intensity of the environment specular when your view direction is parallel to the surface

Translucency: This parameter controls the translucency of raindrops.The surface behand raindrops will become darker when it is edited to a smaller value

Smoothness Scale: This parameter controls how much smoothness will be added to the surface behind raindrops

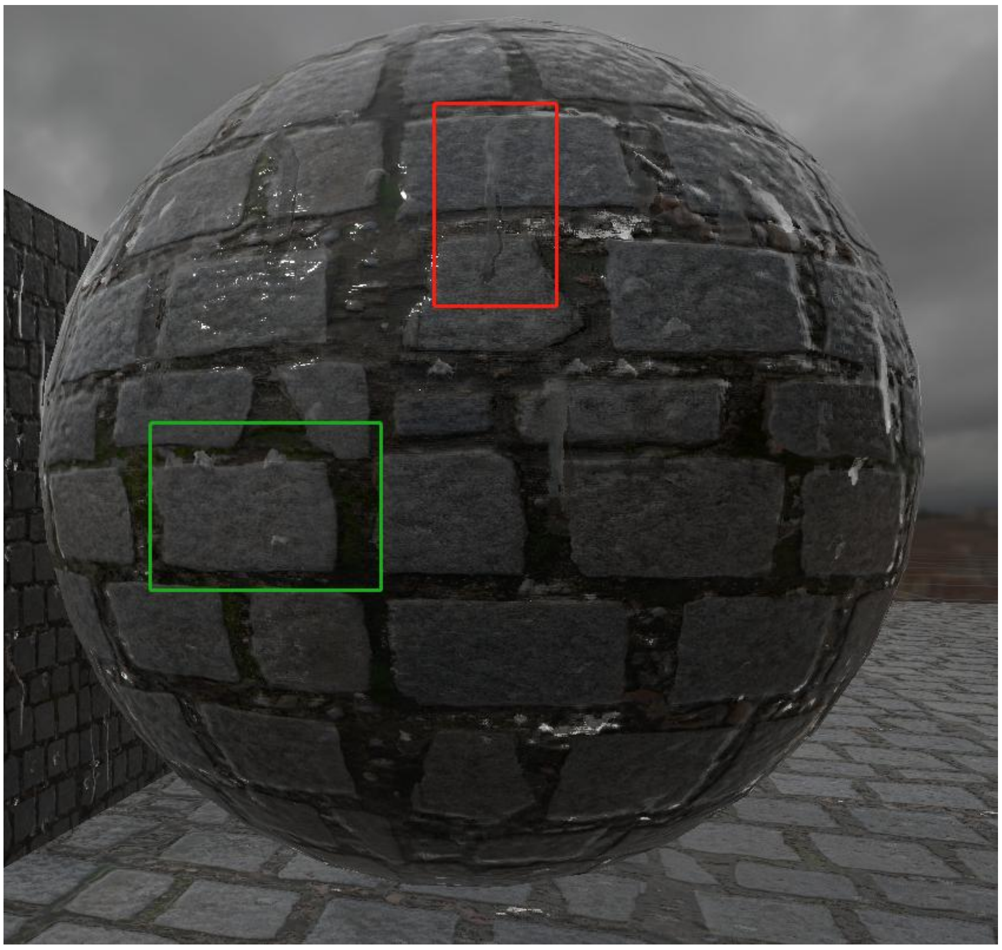
Drop Lighting Scale: This parameter controls the scale of raindrops' lighting

Drop Dot Size: This parameter controls the size of "rain dots",which are some additional wet dots added to the horizontal surface

Drop Dot Repeating: This parameter controls the repeating rate of rain dots. You can also use this parameter to alter the size of rain dots.

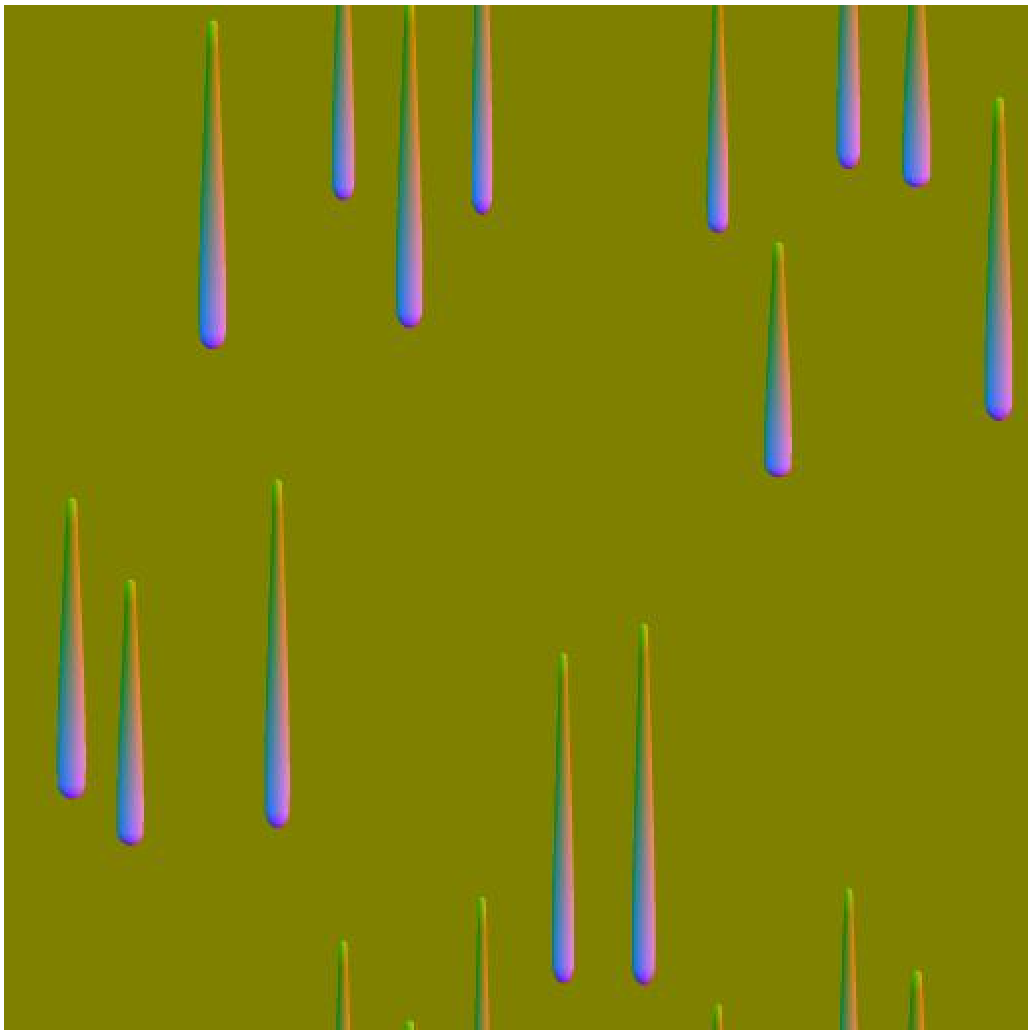
4. Introduction Of Implementation

There are 2 different types of raindrop effect in this package.One is the drops sliding down from objects and the other is the drops hit on the surface. See the red and green part in the image below:



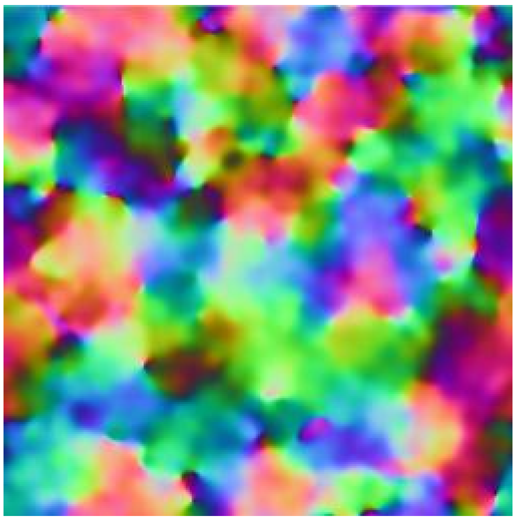
4.1 Drops Sliding Down Effect

First, we generated a texture,which stores raindrops' normal in it's channel RG,and stores the drops mask in the channel B:



This texture is generated by rendering a raindrop mesh in several random places in an orthogonal camera view.You can create your own version of this texture using the CaptureNormal.cs script.

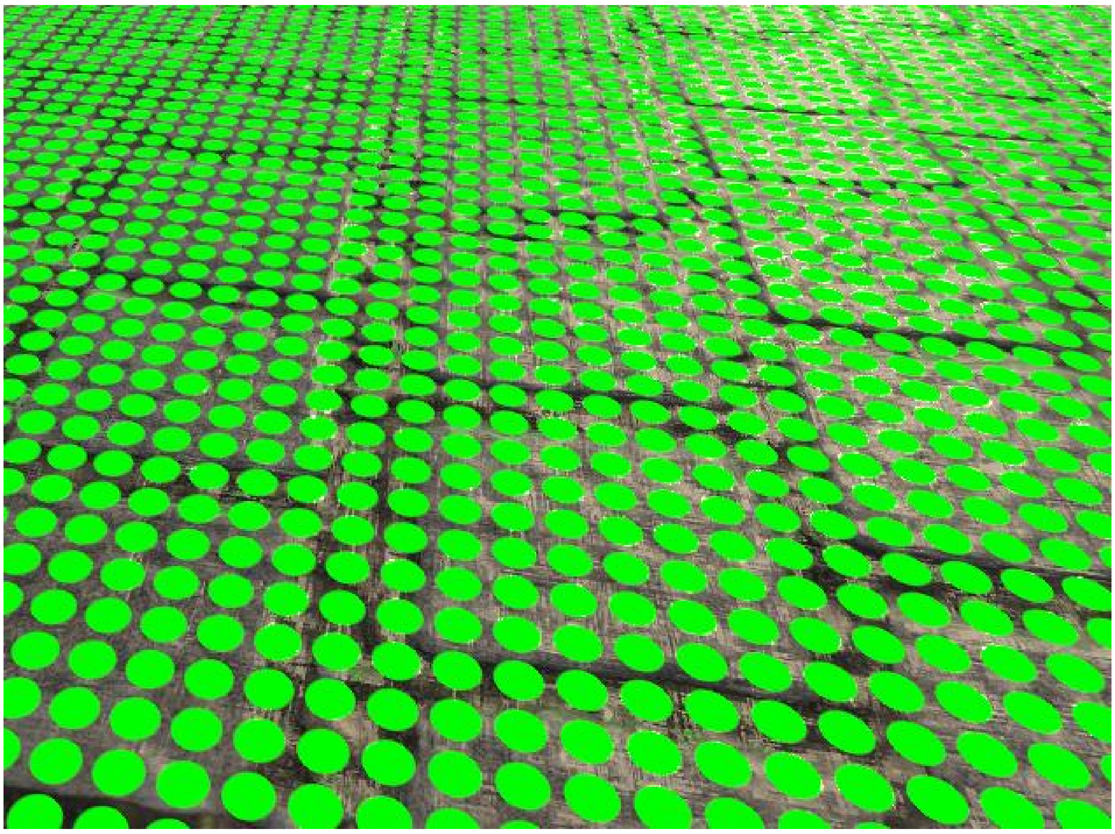
Then in the fragment shader, we sample this texture using the world space position of each pixel as the texture coordinate, and shift it's y axis by time. And use a curl noise to turbulence their moving direction:



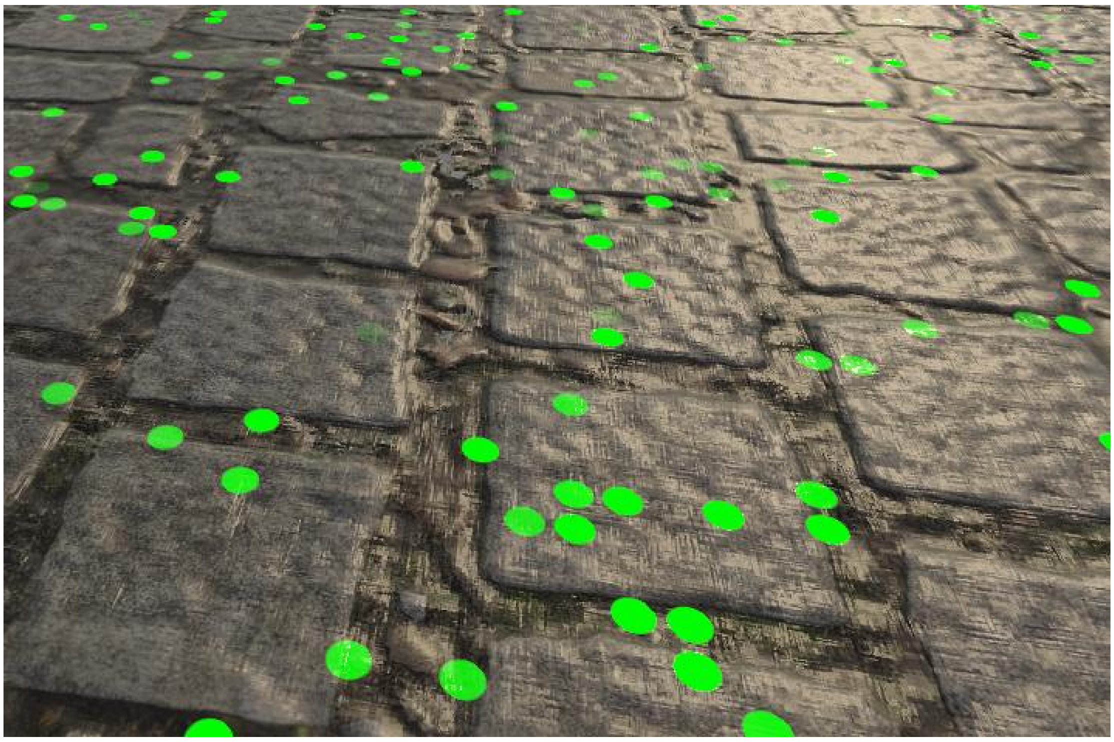
After that,we have already known the intensity and normal of raindrop for each pixel,then we can use those information to add the lighting effects of rain drops.

4.2 Drops Hitting The Surface

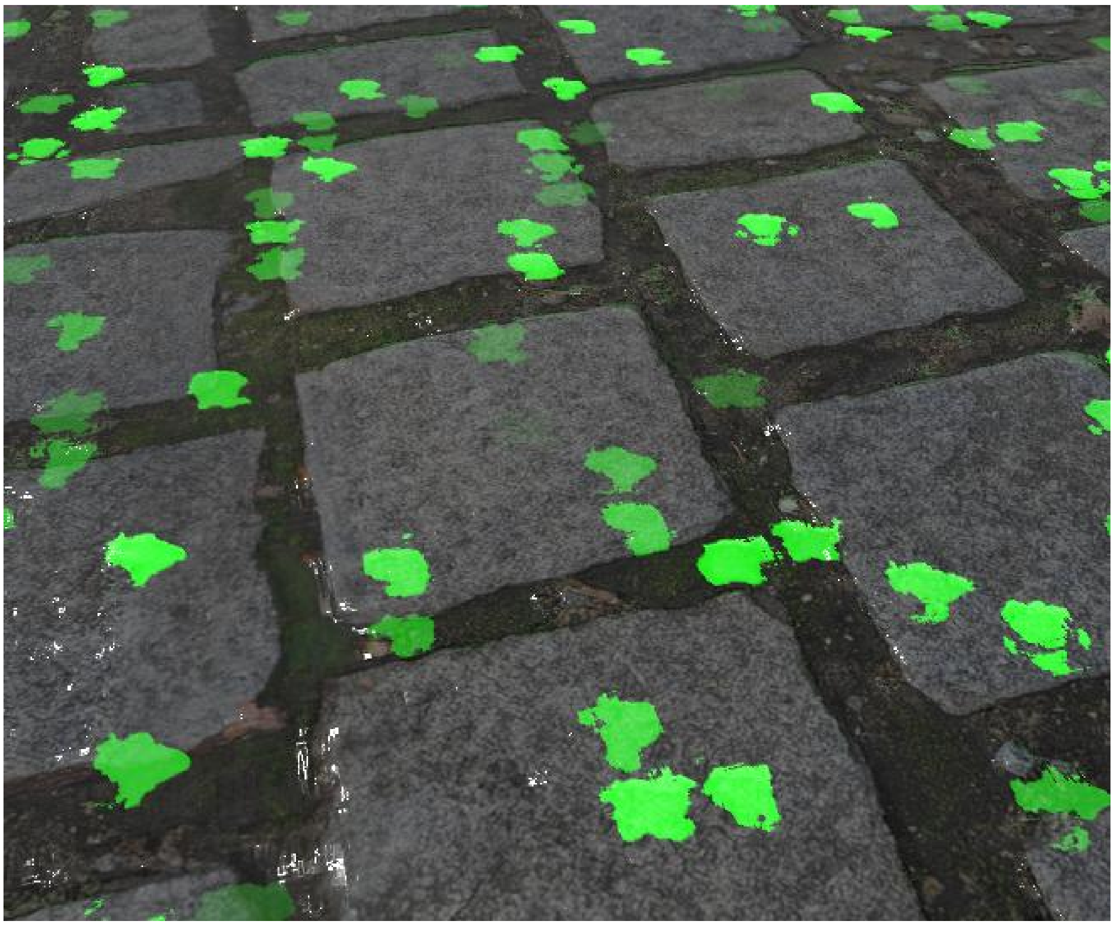
The effect of raindrops hitting on the surface is more simple to implement.First, we generate tiling dots on three axis(showing the xz plane):



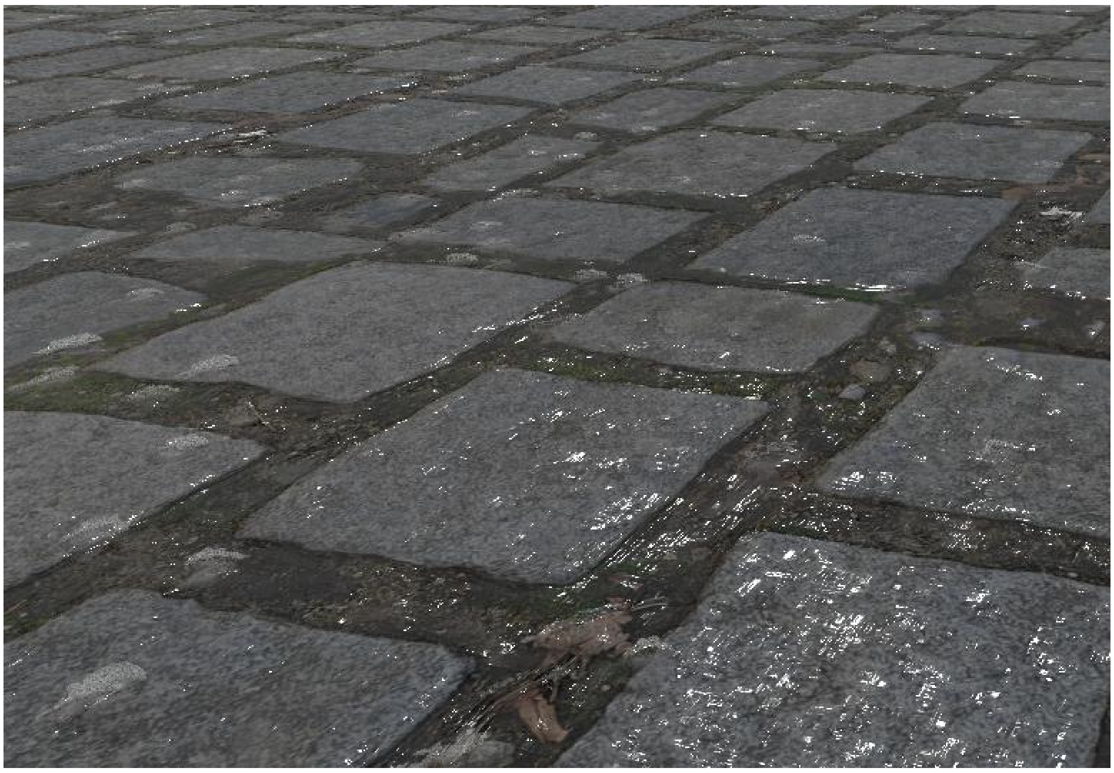
Then we use a noise function shifted by time to mask out some dots,so that they would become more natural:



Distort them:

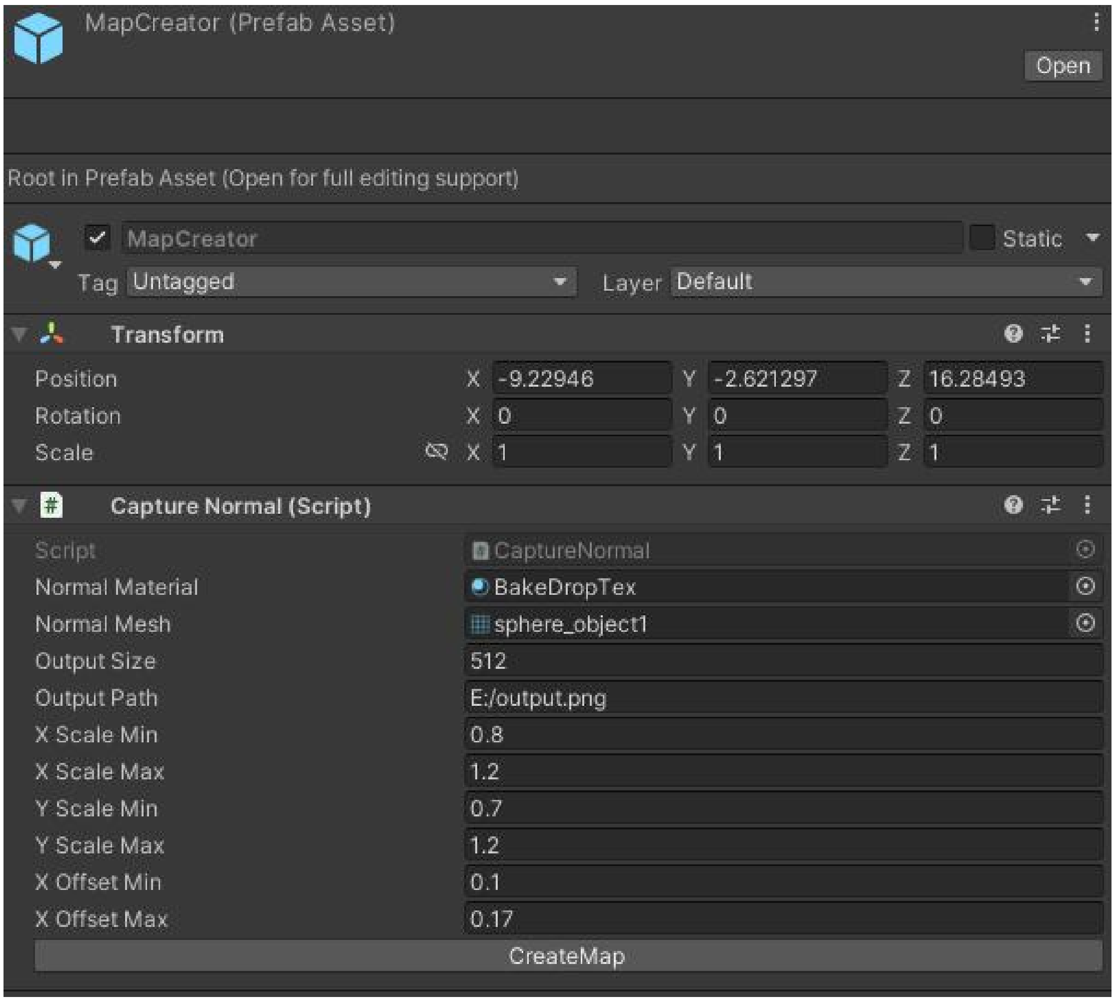


Finally we use this mask to add drop effects like before:



5. Generate Raindrop Texture

You can use the prefab under ‘Prefabs/MapCreator.prefab’ to create the raindrop mask texture:


MapCreator (Prefab Asset)

Open

Root in Prefab Asset (Open for full editing support)

MapCreator Static

Tag Untagged Layer Default

Transform

Position	X	-9.22946	Y	-2.621297	Z	16.28493
Rotation	X	0	Y	0	Z	0
Scale	X	1	Y	1	Z	1

Capture Normal (Script)

Script	CaptureNormal
Normal Material	BakeDropTex
Normal Mesh	sphere_object1
Output Size	512
Output Path	E:/output.png
X Scale Min	0.8
X Scale Max	1.2
Y Scale Min	0.7
Y Scale Max	1.2
X Offset Min	0.1
X Offset Max	0.17

CreateMap

Normal Material: Material used to bake the mask texture,usually you don’t need to change it.

Normal Mesh: The raindrop mesh that would be rendered to the texture.You can use the default mesh or create your own model.

Output Size:Size of the output texture.

Output Path:The path you want the texture to be stored in.

X Scale Min:Low limit of raindrops’ width scale.

X Scale Max:High limit of raindrops’ width scale.

Y Scale Min:Low limit of raindrops’ height scale.

Y Scale Max:High limit of raindrops’ height scale.

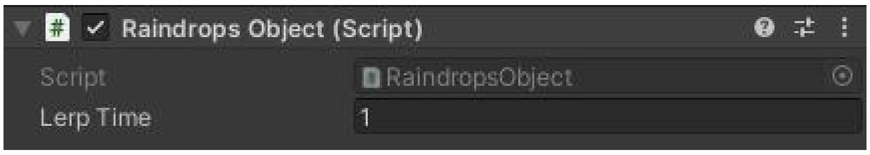
X Offset Min:Low limit of raindrops’ horizontal offset.

X Offset Max:High limit of raindrops’ horizontal offset.

CreateMap:Click this button to render and output the raindrop mask texture.The result texture will be different each time you clicked it, because raindrops will be scaled and placed randomly.

6. Direction Lerp

If you opened the Direction Lerp effect of the material,you should attach the RaindropsObject script to your target GameObject:

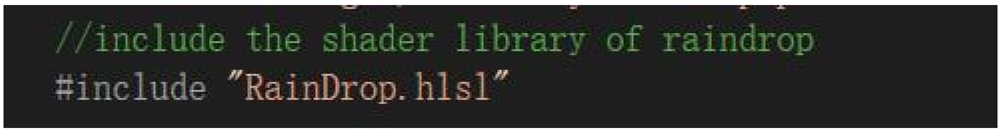


```
Raindrops Object (Script)
Script      RaindropsObject
Lerp Time   1
```

It only has one property ‘Lerp Time’, which is used to control the time interval of the drop direction updating.For example, if your ‘Lerp Time’ is 1 and the global time is 1.2,you will lerp from the direction at time 0 to time 1 in the weight of 0.2.

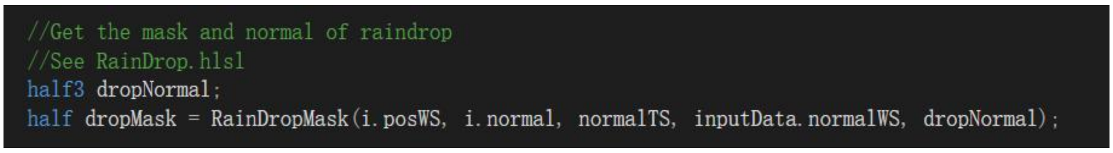
7.Integrating To Your Shader

It’s very easy to integrate those effects into your own shader code.First include the raindrop shader library:



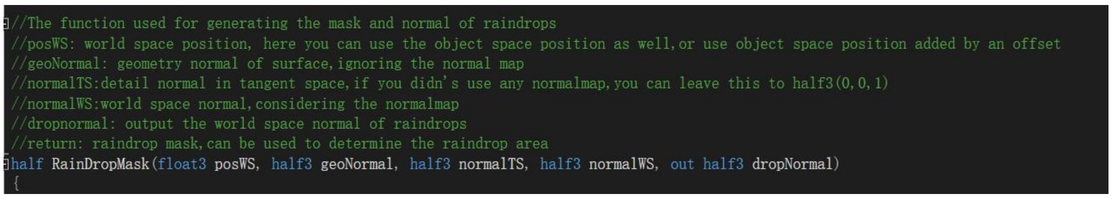
```
//include the shader library of raindrop
#include "RainDrop.hlsl"
```

Then you can call the RainDropMask function to get the normal and mask of raindrops:



```
//Get the mask and normal of raindrop
//See RainDrop.hlsl
half3 dropNormal;
half dropMask = RainDropMask(i.posWS, i.normal, normalTS, inputData.normalWS, dropNormal);
```

Below is the description of this function:



```
//The function used for generating the mask and normal of raindrops
//posWS: world space position, here you can use the object space position as well,or use object space position added by an offset
//geoNormal: geometry normal of surface,ignoring the normal map
//normalTS:detail normal in tangent space,if you didn's use any normalmap,you can leave this to half3(0,0,1)
//normalWS:world space normal,considering the normalmap
//dropnormal: output the world space normal of raindrops
//return: raindrop mask,can be used to determine the raindrop area
half RainDropMask(float3 posWS, half3 geoNormal, half3 normalTS, half3 normalWS, out half3 dropNormal)
{
```

With the given normal and mask information,you can add the raindrops effect by editing the lighting result.You can reference our demo code in RainDrop.shader to see how to do that.