<https://www.baidu.com/link?url=gYeb6BA5Ae-m5S7nRJxP-j6DWm9WRGobJngRqPZ3Ta52UVl4SD9kySi_rm8gHz74&wd=&eqid=97b3e4e80000dee2000000065bd12a19>

前面两种方法都是在shader中直接使用的\_CameraDepthTexture

现在主要是通过c#创建一个深度渲染相机，这个时候就要使用camera的一个大杀器了

**替换渲染Camera.renderwithshader**

它需要两个参数，第一个是替换执行的shader，第二个是决定如何替换规则，一般都是使用RenderType

[ExecuteInEditMode]

[RequireComponent (typeof(Camera))]

public class LightDepthTexture : MonoBehaviour {

[HideInInspector]

public RenderTexture targetTexture;

public GameObject lightObj;

public Shader shader;

private Material myMaterial;

private Material material

{

get

{

if (!myMaterial) myMaterial = new Material(shader);

return myMaterial;

}

}

void OnRenderImage(RenderTexture src, RenderTexture dest)

{

if (material)

{

Camera depthCamera = lightObj.GetComponent<Camera>();

if (!depthCamera) depthCamera = lightObj.AddComponent<Camera>();

depthCamera.orthographic = true; // 设置是否是正交相机

depthCamera.enabled = false;

depthCamera.clearFlags = CameraClearFlags.Depth;

depthCamera.hideFlags = HideFlags.HideAndDontSave;

int width = Screen.width;

int height = Screen.height;

// 暂时把RenderDepth放在设置中的always include shaders中

Shader replaceShader = Shader.Find("Unlit/RenderDepth");

Debug.Log(replaceShader);

targetTexture = new RenderTexture(width, height, 0);

depthCamera.targetTexture = targetTexture;

// replacementTag如果为空，则所有物体都用该replaceShader，

// 如果是RenderType，则会替换物体shader中RenderType和replaceShader中RenderType相等的物体

depthCamera.RenderWithShader(replaceShader, "");

material.SetTexture("\_DepthMap", targetTexture);

Graphics.Blit(src, dest, material);

// 激活渲染贴图读取信息

RenderTexture.active = targetTexture; // 如果不添加这行则仍然是从屏幕中读取的

Texture2D texture = new Texture2D(width, height, TextureFormat.RGB24, false);

texture.ReadPixels(new Rect(0, 0, width, height), 0, 0);

texture.Apply();

byte[] bytes = texture.EncodeToPNG();

System.IO.File.WriteAllBytes(Application.dataPath + "/a.png", bytes);

RenderTexture.active = null;

targetTexture = null;

depthCamera.targetTexture = null;

}

else

{

Graphics.Blit(src, dest);

}

}

}

**注意：如果我们想用当前正在渲染的相机来渲染一遍场景就会出现警告提示**

****

**这时候我们就需要重新创建一个Camera，然后CopyFrom当前的Camera即可  
Camera camera = obj.AddComponent<Camera>();**

**camera.CopyFrom(gameObject.GetComponent<Camera>());**

**camera.clearFlags = CameraClearFlags.Depth;**

**camera.enabled = false;**

产生depth的shader：

Shader "Unlit/RenderDepth"

{

SubShader

{

Tags { "RenderType"="Opaque"}

Pass

{

CGPROGRAM

#pragma vertex vert

#pragma fragment frag

#include "UnityCG.cginc"

struct appdata

{

float4 vertex : POSITION;

};

struct v2f

{

float4 vertex : SV\_POSITION;

};

v2f vert (appdata v)

{

v2f o;

o.vertex = UnityObjectToClipPos(v.vertex);

return o;

}

fixed4 frag (v2f i) : SV\_Target

{

return fixed4(i.vertex.z, i.vertex.z, i.vertex.z, 1.0);

}

ENDCG

}

}

}

显示depth的shader：

Shader "Unlit/LightDepthTexture"

{

Properties

{

\_MainTex ("Texture", 2D) = "white" {}

\_DepthMap ("Depth Map", 2D) = "white" {}

}

SubShader

{

Tags { "RenderType"="Opaque" }

Pass

{

CGPROGRAM

#pragma vertex vert

#pragma fragment frag

#include "UnityCG.cginc"

struct appdata

{

float4 vertex : POSITION;

float2 uv : TEXCOORD0;

};

struct v2f

{

float2 uv : TEXCOORD0;

float2 uv\_depth : TEXCOORD1;

float4 vertex : SV\_POSITION;

};

sampler2D \_MainTex;

float4 \_MainTex\_ST;

float4 \_MainTex\_TexelSize;

sampler2D \_DepthMap;

float4 \_DepthMap\_ST;

v2f vert (appdata v)

{

v2f o;

o.vertex = UnityObjectToClipPos(v.vertex);

o.uv = TRANSFORM\_TEX(v.uv, \_MainTex);

o.uv\_depth = TRANSFORM\_TEX(v.uv, \_DepthMap);

#if UNITY\_UV\_STARTS\_AT\_TOP

if (\_MainTex\_TexelSize.y < 0) o.uv\_depth.y = 1 - o.uv\_depth.y;

#endif

return o;

}

fixed4 frag (v2f i) : SV\_Target

{

fixed4 col = tex2D(\_DepthMap, i.uv\_depth);

return col;

}

ENDCG

}

}

}



