<https://blog.csdn.net/quan2008happy/article/details/39380463>

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);

}

}