In this demo, I realized several advanced effects by using the unity3d engine. I programed all the code and shader for the demo, the art resources came from some public resources on the Internet. The Demo is showed as a unity project, which was developed in Unity3D 5.4 version. There are three scenes, the details of which will be listed as follows:

1.Shadowmap

The scene file is UnityEffects\Assets\Scene\(1)ShadowMap.unity

1. I use ShadowMap technology to create shadow effects in the scene. ShadowMap is a popular shadow technology, which is widely used in various game engines. The first step is using \_CameraDepthTexture in unity3D to generate the depth map of the scene. Then, when rendering, the depth of the pixel to be rendered is compared to the depth of the corresponding position in the depth map to determine whether the pixel is in shadow.
2. The C# code for these functions is mainly in the files:

UnityEffects\Assets\Script\(1)ShadowMap\CreateDepthMap.cs

UnityEffects\Assets\Script\(1)ShadowMap\ShadowMap.cs

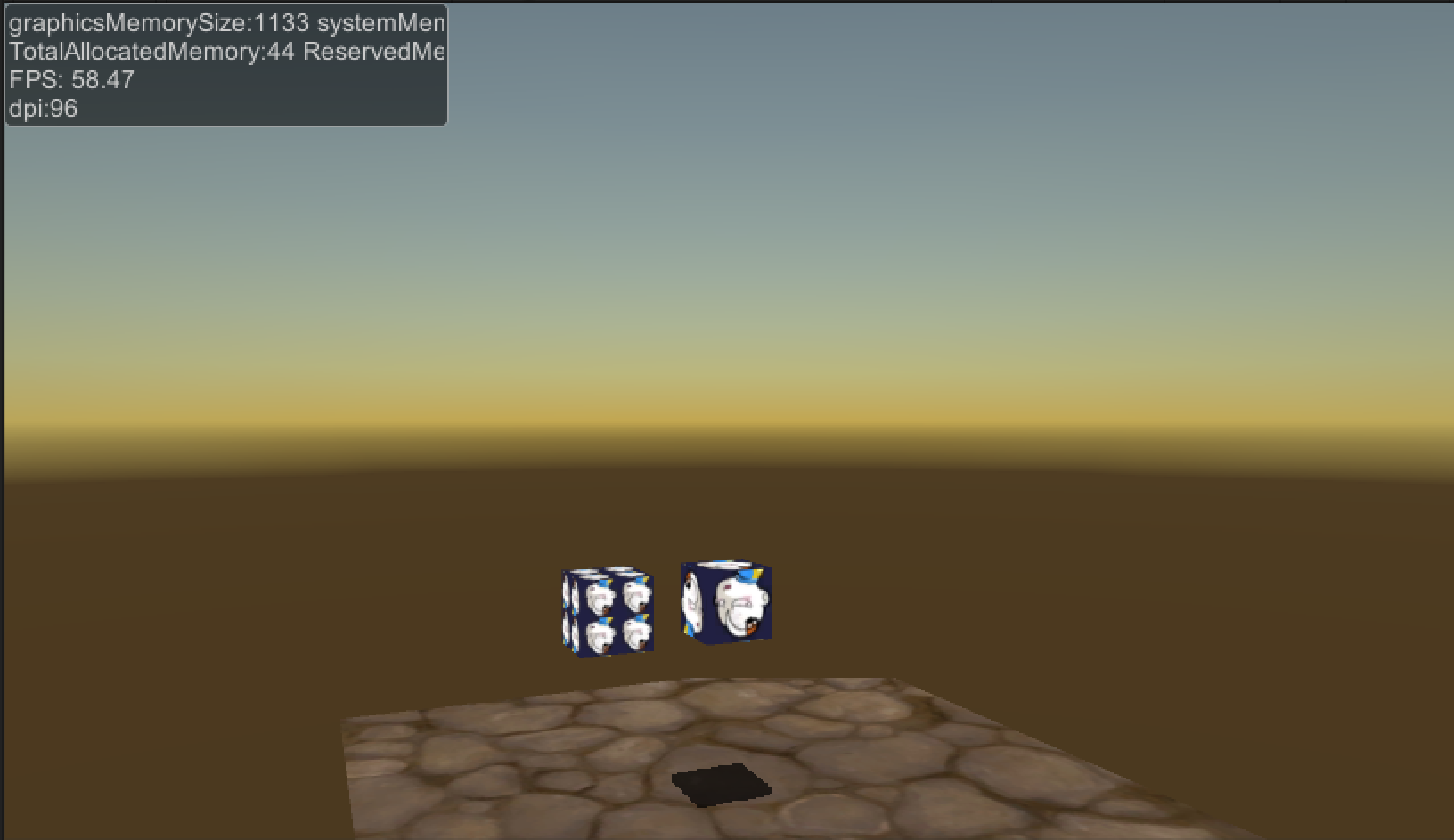
1. The shader code for rendering is mainly in files:

UnityEffects\Assets\ShaderAndMat\(1)ShadowMap\DepthMap.shader

UnityEffects\Assets\ShaderAndMat\(1)ShadowMap\DepthMap2.shader

UnityEffects\Assets\ShaderAndMat\(1)ShadowMap\ShadowMap.shader

UnityEffects\Assets\ShaderAndMat\(1)ShadowMap\ ShadowMapUnlit.shader

1. CreateDepthMap.cs is attached to an Gameobject named LightCamera and cooperates with DepthMap.shader to render the depth of the scene into a depth texture. DepthMap2.shader provides an alternative method of depth calculation. ShadowMap.cs then is attached on the object receiving shadow and renders the object with ShadowMap.shader. ShadowMap.cs is used to pass parameters of the LightCamera. ShadowMap.shader draws the shadow effect correctly according to the result of depth comparison.
2. The effect is as follow:  
   

Information about this scene are often shared in my blog:

https://blog.csdn.net/aceyan0718/article/details/52067264

2.ProjectorShadow

The scene file is UnityEffects\Assets\Scene\(2)ProjectorShadow.unity

1. In this scene, I use the unity3D Projector to implement the shadow effect, which is an alternative to ShadowMap with higher performance and is more suitable for implementing simple shadow effect in the mobile platform. And this method has been successfully used in several mobile game projects developed by me in the company where I worked before.
2. The C# code for these functions is mainly in the files:

UnityEffects\Assets\Script\(2)ProjectorShadow\ShadowProjector.cs

1. The shader code for rendering is mainly in files:

UnityEffects\Assets\ShaderAndMat\(2)ProjectorShadow\ShadowProjector.shader

UnityEffects\Assets\ShaderAndMat\(2)ProjectorShadow\ShadowReplace.shader

1. In the scene file, the ShadowProjector.cs has been attached to a Projector Gameobject with a Camera component. Then I use the SetReplacementShader function to render the shadow casters’ (objects casting shadow) ‘profile’ into a texture. The texture was then projected into the shaded object surface as the shadow by using the projector and ShadowProjector. Shader.
2. The effect is as follow:



We can notice that this method can also generate transparent objects’ shadow perfectly.

Information about this scene are often shared in my blog:

https://blog.csdn.net/aceyan0718/article/details/52279594

3. ChainLightning

The scene file is UnityEffects\Assets\Scene\(3)ChainLightning.unity

1. By using algorithm of midpoint displacement and UV animation, I simulate lightning effects in this scene. Algorithm of midpoint below is a approach to produce subdivisions on segments and it is a great way to generate shape of lightning. UV animation is used to create the sensation of lightning flowing.
2. The c# code for these functions is mainly in the files:

UnityEffects\Assets\Script\(3)ChainLightning\UVChainLightning.cs

1. This approach is also widely used in my companies’ mobile games that are already online.
2. The effect is as follow:



Information about this scene are often shared in my blog:

https://blog.csdn.net/aceyan0718/article/details/52317231