<https://blog.csdn.net/puppet_master/article/details/52889050>

Bloom特效时游戏中常见的一种屏幕特效。这种特效可以模糊真实摄像机的一种图像效果，它让画面中较亮的区域“扩散”到周围的区域中，造成一种朦胧效果。

Bloom的实现原理非常简单：我们首先根据一个阈值提取出图像中较亮区域，把他们存储在一张渲染纹理中，再利用高斯模糊对这张渲染纹理进行模糊处理，模拟光线扩散的效果，最后再将其和原图像进行混合，得到最终的效果。

Shader "Unlit/Bloom"

{

Properties

{

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

\_Bloom ("Bloom", 2D) = "black" {}

\_LuminanceThreshold ("Luminance Threshold", Float) = 0.5

\_BlurSize ("Blur Size", Float) = 1.0

}

SubShader

{

ZTest Always

Cull Off

ZWrite Off

CGINCLUDE

#include "UnityCG.cginc"

sampler2D \_MainTex;

float4 \_MainTex\_TexelSize;

sampler2D \_Bloom;

float \_LuminanceThreshold;

float \_BlurSize;

struct v2f

{

float4 pos : SV\_POSITION;

half2 uv : TEXCOORD0;

};

v2f vertExtractBright(appdata\_img v)

{

v2f o;

UNITY\_INITIALIZE\_OUTPUT(v2f, o);

o.pos = UnityObjectToClipPos(v.vertex);

o.uv = v.texcoord;

return o;

}

fixed luminance(fixed4 color)

{

return 0.2125 \* color.r + 0.7154 \* color.g + 0.0721 \* color.b;

}

fixed4 fragExtractBright(v2f o) : SV\_Target

{

fixed4 c = tex2D(\_MainTex, o.uv);

fixed4 val = clamp(luminance(c) - \_LuminanceThreshold, 0.0, 1.0);

return c \* val;

}

struct v2fBloom

{

float4 pos : SV\_POSITION;

half4 uv : TEXCOORD0;

};

v2fBloom vertBloom(appdata\_img v)

{

v2fBloom o;

UNITY\_INITIALIZE\_OUTPUT(v2fBloom, o);

o.pos = UnityObjectToClipPos(v.vertex);

o.uv.xy = v.texcoord;

o.uv.zw = v.texcoord;

#if UNITY\_UV\_STARTS\_AT\_TOP

if (\_MainTex\_TexelSize.y < 0.0)

o.uv.w = 1.0 - o.uv.w;

#endif

return o;

}

fixed4 fragBloom(v2fBloom o) : SV\_Target

{

return tex2D(\_MainTex, o.uv.xy) + tex2D(\_Bloom, o.uv.zw);

}

ENDCG

Pass

{

CGPROGRAM

#pragma vertex vertExtractBright

#pragma fragment fragExtractBright

ENDCG

}

UsePass "Unlit/GaussianBlur/GAUSSIAN\_BLUR\_VERTICAL"

UsePass "Unlit/GaussianBlur/GAUSSIAN\_BLUR\_HORIZONTAL"

Pass

{

CGPROGRAM

#pragma vertex vertBloom

#pragma fragment fragBloom

ENDCG

}

}

}

C#代码：

void OnRenderImage(RenderTexture src, RenderTexture dest)

{

if (material)

{

material.SetFloat("\_LuminanceThreshold", luminanThreshold);

int rtW = src.width / downSample;

int rtH = src.height / downSample;

RenderTexture buffer0 = RenderTexture.GetTemporary(src.width, src.height, 0);

buffer0.filterMode = FilterMode.Bilinear;

Graphics.Blit(src, buffer0, material, 0); // 第一个pass，提取较亮区域

for (int i = 0; i < iterations; i++)

{

material.SetFloat("\_BlurSize", 1.0f + blurSpread);

RenderTexture buffer1 = RenderTexture.GetTemporary(rtW, rtH, 0);

Graphics.Blit(buffer0, buffer1, material, 1); // 第二个pass

RenderTexture.ReleaseTemporary(buffer0);

buffer0 = buffer1;

buffer1 = RenderTexture.GetTemporary(rtW, rtH, 0);

Graphics.Blit(buffer0, buffer1, material, 2); // 第三个pass

RenderTexture.ReleaseTemporary(buffer0);

buffer0 = buffer1;

}

material.SetTexture("\_Bloom", buffer0);

Graphics.Blit(src, dest, material, 3); // 第四个pass

RenderTexture.ReleaseTemporary(buffer0);

}

else

{

Graphics.Blit(src, dest);

}

}