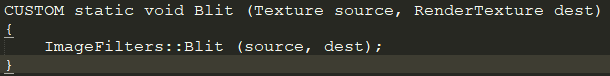
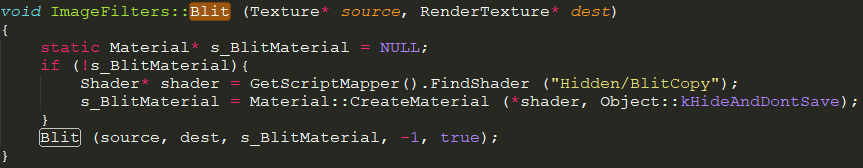
查看4.3.1源码：





从这里可以看出，如果设置了material，则使用设置的material，如果没有设置，则通过shader BlitCopy创建一个material使用

void ImageFilters::Blit (Texture\* source, RenderTexture\* dest, Unity::Material\* mat, int pass, bool setRT)

{

using namespace ImageFilters\_Static;

PROFILER\_AUTO(gGraphicsBlitProfile, mat->GetShader())

GfxDevice& device = GetGfxDevice();

UInt32 rtFlags = 0;

#if UNITY\_XENON

// Xbox 360 must resolve a render target before using it as a texture.

if (source == dest)

{

rtFlags |= RenderTexture::kFlagForceResolve;

}

else if (!setRT)

{

// Render target was set previously. Get it and compare.

if (source == device.GetActiveRenderTexture())

{

setRT = true;

rtFlags |= RenderTexture::kFlagForceResolve;

}

}

#endif

// MSAA render targets must be resolved before they are used.

if (IsActiveRenderTextureMSAA())

{

setRT = true;

rtFlags |= RenderTexture::kFlagForceResolve;

}

if (setRT)

SetCurrentRenderTarget (dest, rtFlags);

bool setTexture = source && mat->HasProperty(kSLPropMainTex);

**if (setTexture)**

**mat->SetTexture (kSLPropMainTex, source); // 设置\_MainTex属性**

bool invertY = source && source->GetTexelSizeY() < 0.0f;

float uvX = 1.0f, uvY = 1.0f;

#if GFX\_EMULATES\_NPOT\_RENDERTEXTURES

if (source)

{

int texWidth = source->GetGLWidth();

int texHeight = source->GetGLHeight();

uvX = (float)texWidth / (float)NextPowerOfTwo(texWidth);

uvY = (float)texHeight / (float)NextPowerOfTwo(texHeight);

}

#endif

DeviceMVPMatricesState preserveMVP;

**LoadFullScreenOrthoMatrix(); // 设置正交相机的mvp矩阵**

int npasses = mat->GetPassCount ();

if (pass == -1) **// 如果pass==-1，则执行所有的pass，否则执行特定pass**

{

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

{

mat->SetPass (i);

**DrawQuad (device, invertY, uvX, uvY); // 绘制四边形**

}

}

else

{

if (pass >= 0 && pass < npasses)

{

mat->SetPass (pass);

**DrawQuad (device, invertY, uvX, uvY); // 绘制四边形**

}

else

{

ErrorString ("Invalid pass number for Graphics.Blit");

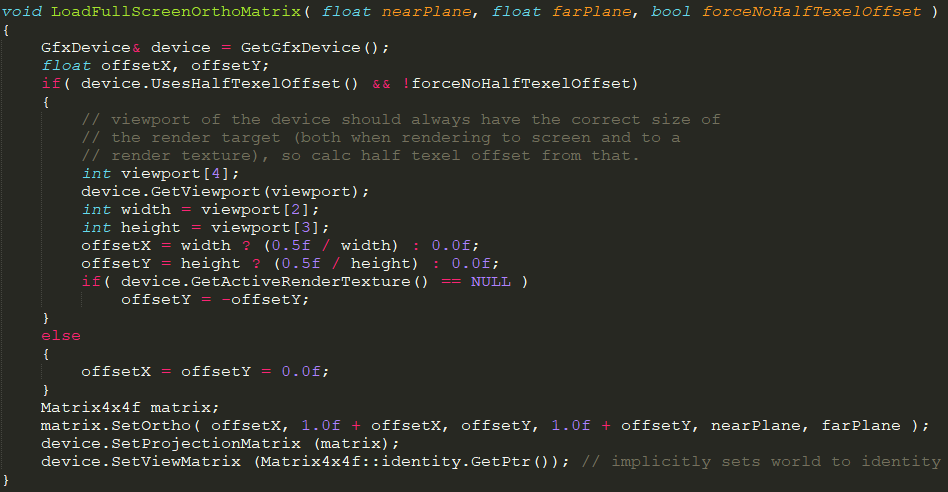
}

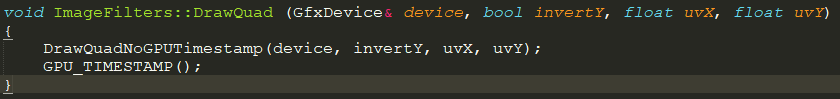
}

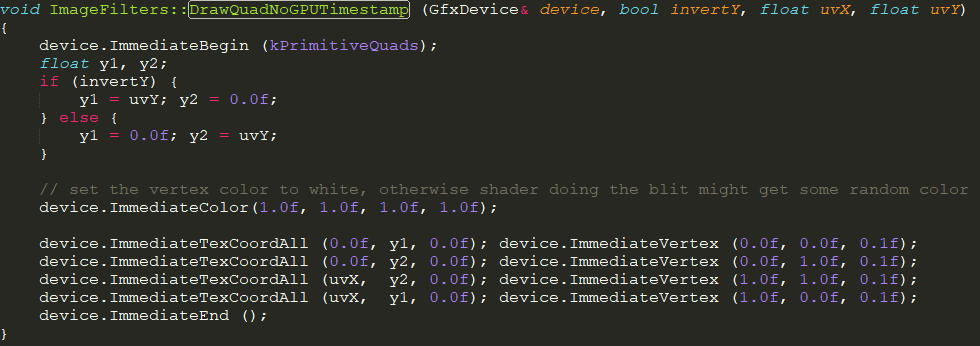
if (setTexture)

mat->SetTexture (kSLPropMainTex, NULL);

}

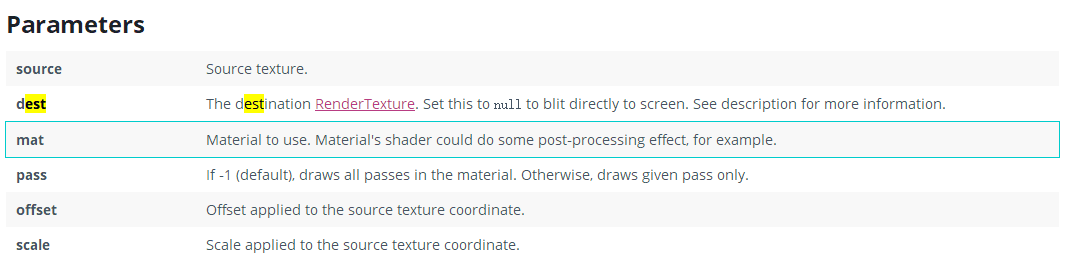






实际上就是画了一个正方形，采用渲染到纹理的技术，最后将纹理以正方形的形式画出

两次渲染  
第一次正常渲染到纹理（屏幕外）  
第二次将纹理正常渲染到默认帧缓冲（屏幕内）



**看参数可以知道，如果dest为Null，则表示直接渲染到屏幕**