<https://blog.csdn.net/fenglele_fans/article/details/79242351>

单个相机截屏

局部截屏

Unity的三种截屏方式

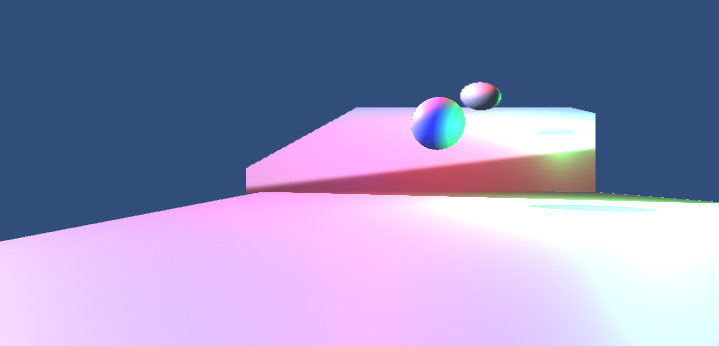
**方式1）系统全屏截屏**

void CaptureUnity(string filePath)

{

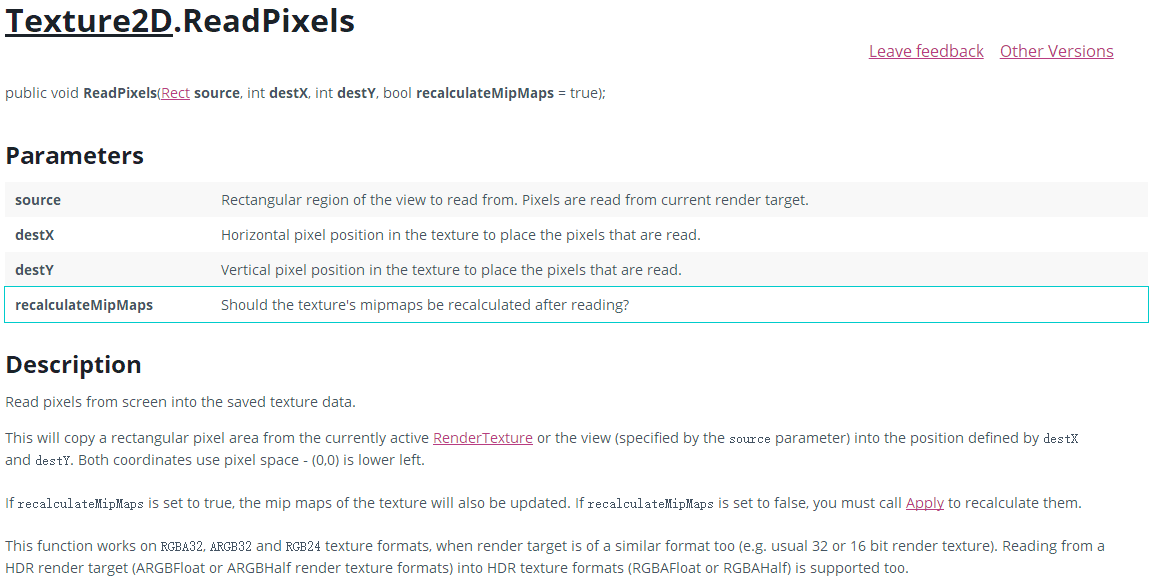
Application.CaptureScreenshot(filePath);

}



这种方式最简单，确定也很明显，比如不能选择区域，不能选择图片格式，不能屏蔽某些对象

**方式2）根据Rect类型来截取指定范围的屏幕**



IEnumerator CaptureScreen()

{

yield return new WaitForEndOfFrame();

CaptureByRect(new Rect(100, 50, 400, 200), Application.dataPath + "/t.png");

}

void CaptureByRect(Rect rect, string filePath)

{

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

texture.ReadPixels(rect, 0, 0, false);

texture.Apply();

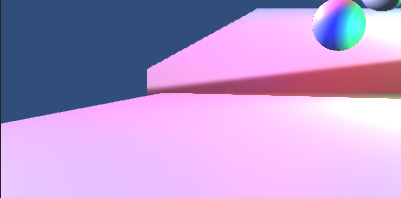
byte[] bytes = texture.EncodeToPNG();

System.IO.File.WriteAllBytes(filePath, bytes);

}

**赋值完以后为什么要texture2D.apply()?**

因为贴图更改像素时并不是直接对显存进行更改，而是在另外一个内存空间中更改，这时GPU还会实时读取旧的贴图。当Apply后，CPU会告诉GPU，你要换个地方读取贴图了。



**方式3）截取多个Camera渲染**

void CaptureByCamera(Camera camera, Rect rect, string filePath)

{

// 初始化RenderTexture

RenderTexture renderTexture = new RenderTexture((int)rect.width, (int)rect.height, 0);

// 设置相机的渲染目标

camera.targetTexture = renderTexture;

// 渲染

camera.Render();

// 这里可以同时渲染多个相机

Camera camera1 = GameObject.Find("Camera1").GetComponent<Camera>();

camera1.targetTexture = renderTexture;

camera1.Render();

// 激活渲染贴图读取信息

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

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

texture.ReadPixels(rect, 0, 0);

texture.Apply();

//释放相机，销毁渲染贴图

camera.targetTexture = null;

camera1.targetTexture = null;

RenderTexture.active = null;

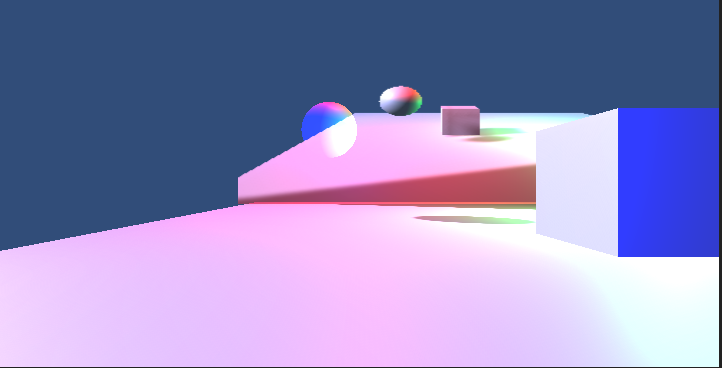
GameObject.Destroy(renderTexture);

renderTexture.Release();

byte[] bytes = texture.EncodeToPNG();

System.IO.File.WriteAllBytes(filePath, bytes);

}



其中蓝色的立方体是camera1看到的，其余的是camera看到的