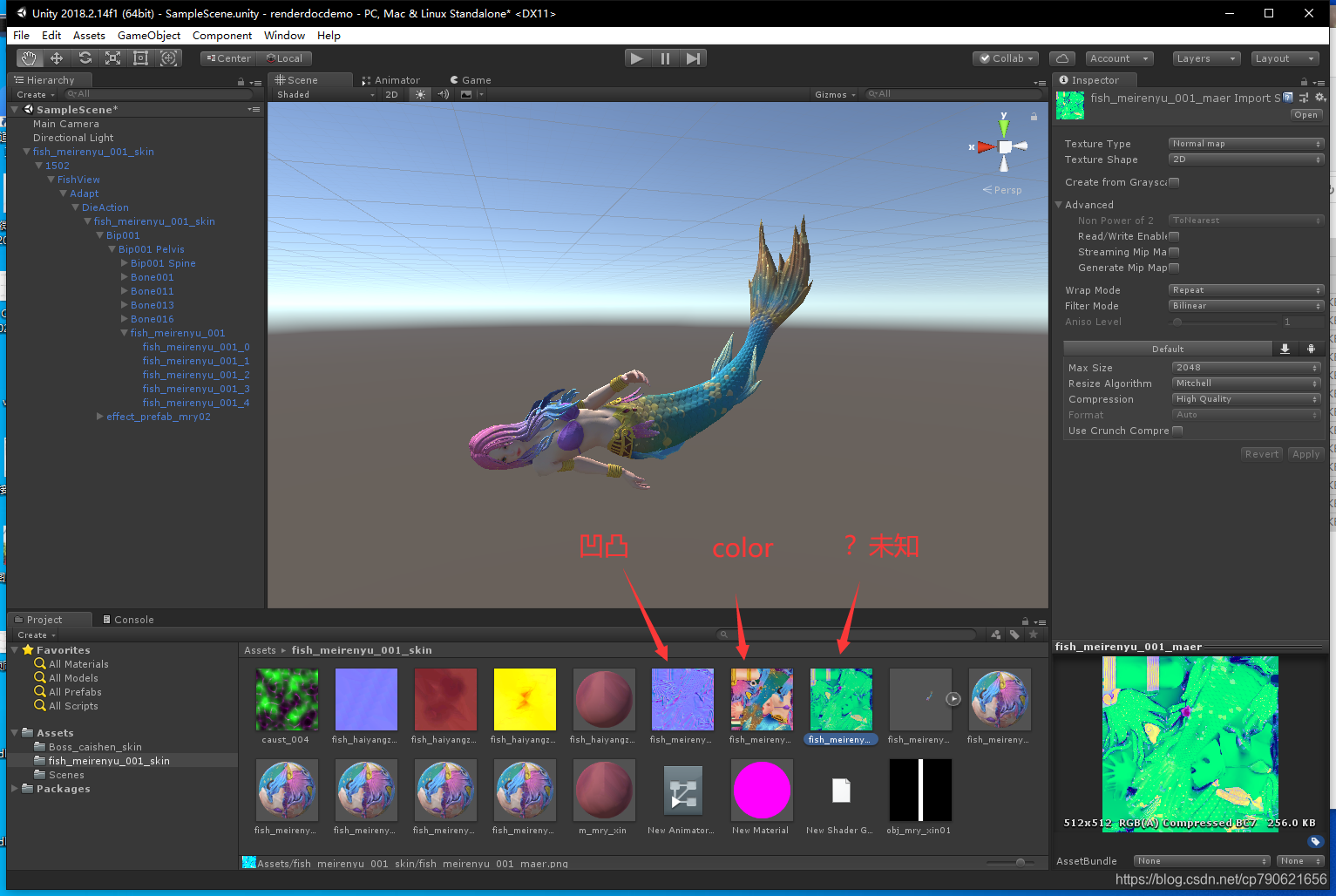
1.背景

这几天玩了一个美术表现不错的游戏，忍不住想扒点模型资源和效果。

直接用UnityStudio解包拿到fbx。

但是一起解压出来的三张贴图，一张普通贴图，一张法线，还有一张就难住我了。



这个绿色的是什么用处的，在群里问了一圈小伙伴也表示不知道？

没有什么头绪，那就只好用GPU分析软件来调试。

上一次用GPU分析软件还是2年前了，用的是Mali的MGD，需要搭配Root的手机。

现在手上只有一台自用红米Note8，没有Root，于是本次使用群大佬推荐的RenderDoc。

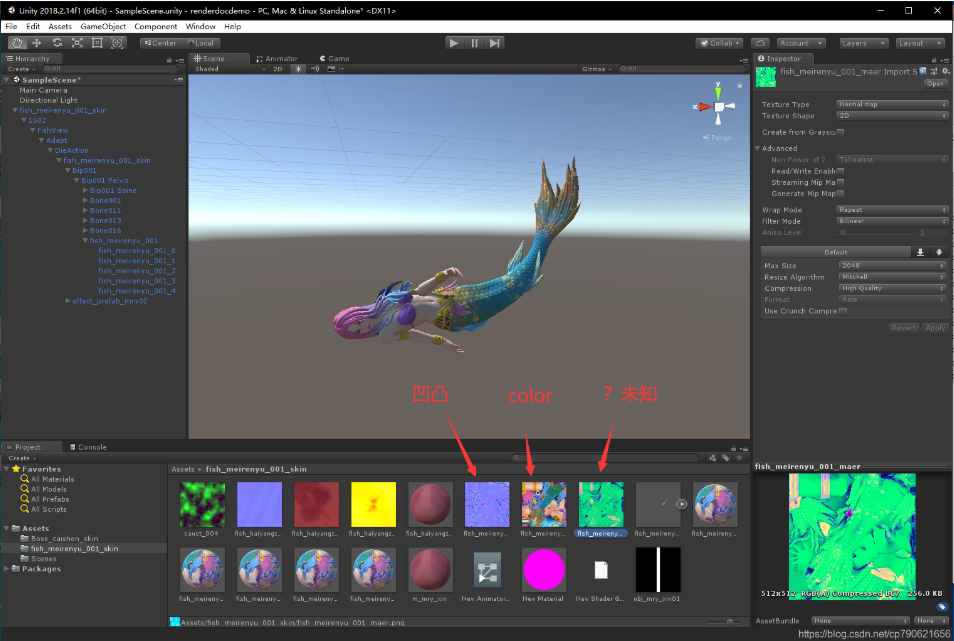
本文转自https://huutu.blog.csdn.net/article/details/111659729

RenderDoc之前我只在PC上用过，没想到对Android支持也如此完善，有多完善，往下看就知道。

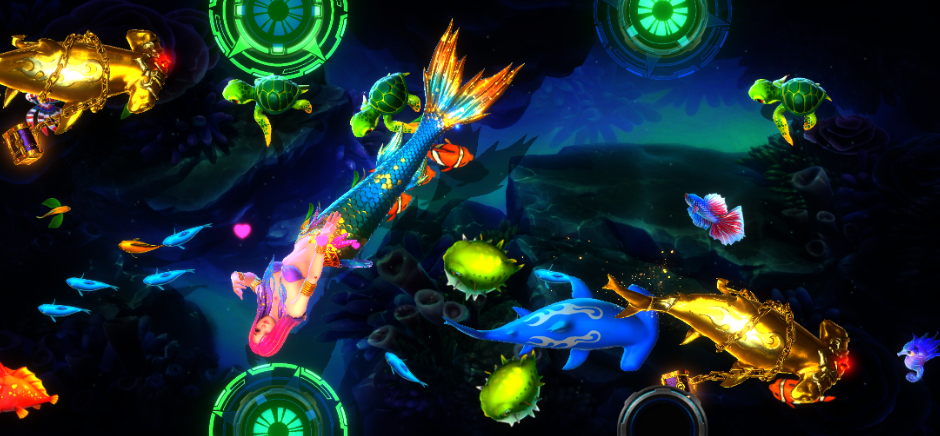
2.目标

这次分析，主要解决3个疑问：

1. 美人鱼的三张纹理中，绿色的那张是什么用途？



2.游戏中bloom的方式



3.冰冻特效如何制作



3.RenderDoc调试分析基础

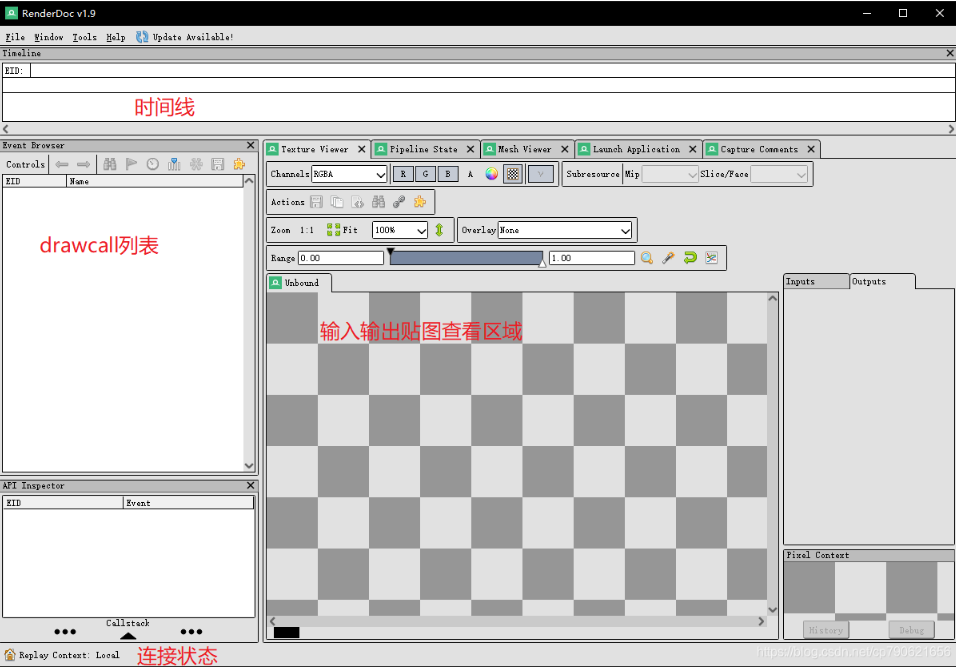
官网：https://renderdoc.org/

GameDevTools已经集成，搜索 RenderDoc即可下载运行。

GameDevTools是一套游戏开发常用工具集，输入关键词即可找到想要的工具。

目前托管在Github：https://github.com/ThisisGame/GameDevTools

安装后打开，界面如下：本文转自https://huutu.blog.csdn.net/article/details/111659729



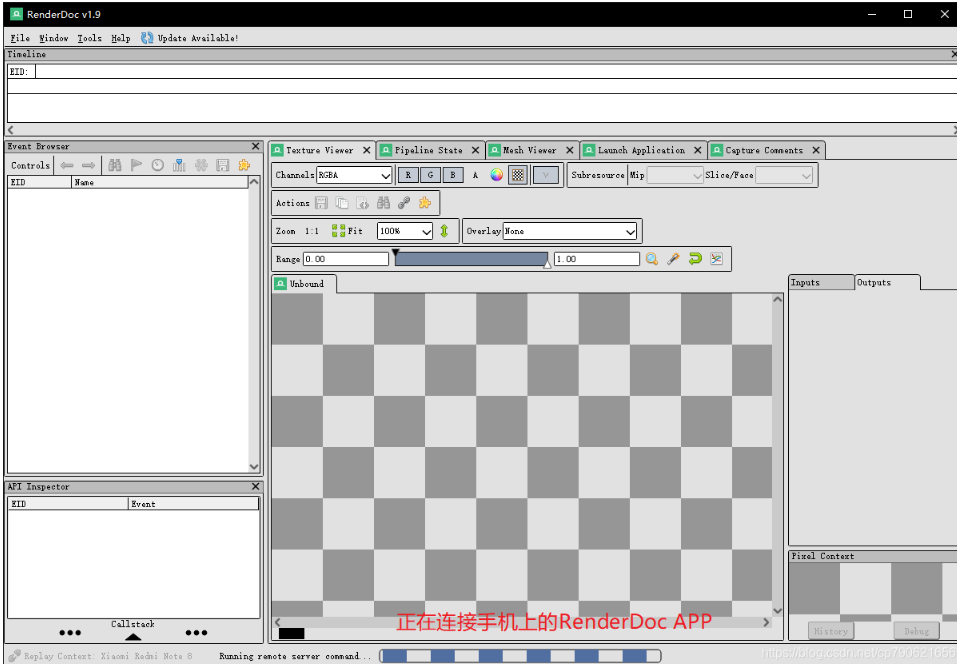
3.1 连接Android手机(无需root)

手机开启调试模式，打开AndroidStudio看logcat是否正常输出Log，确认连接OK。

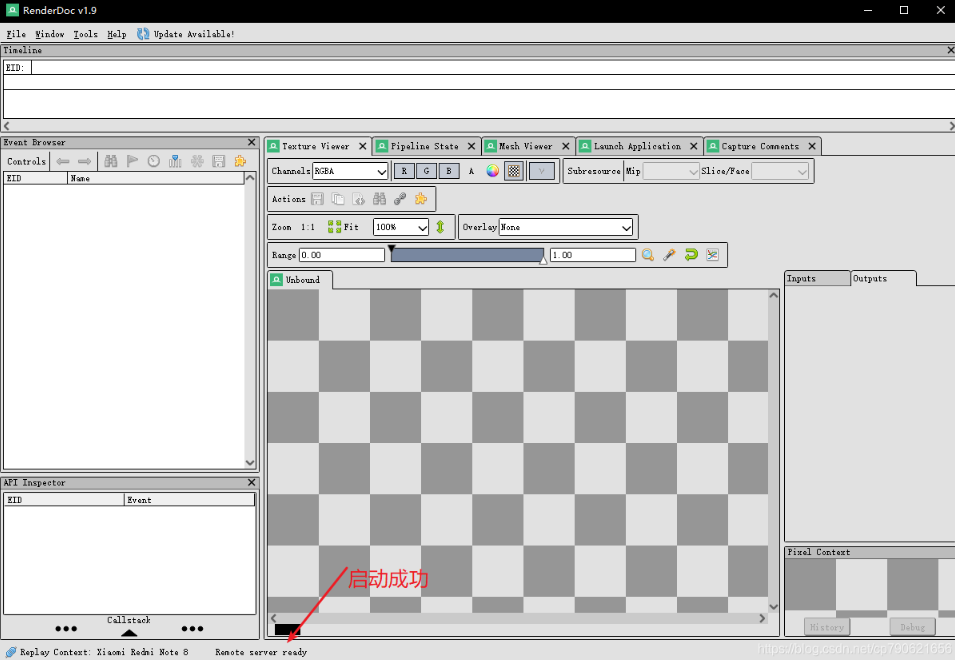
点击RenderDoc左下角，切换到手机进行连接。

选择手机之后，会通过adb，往手机上安装一个RenderDocCmd的插件app，然后启动。

RenderDocCmd hook了Android的 gles渲染接口，然后将数据回传到PC端的RenderDoc。



上图表示正在启动手机上的RenderDocCmd。

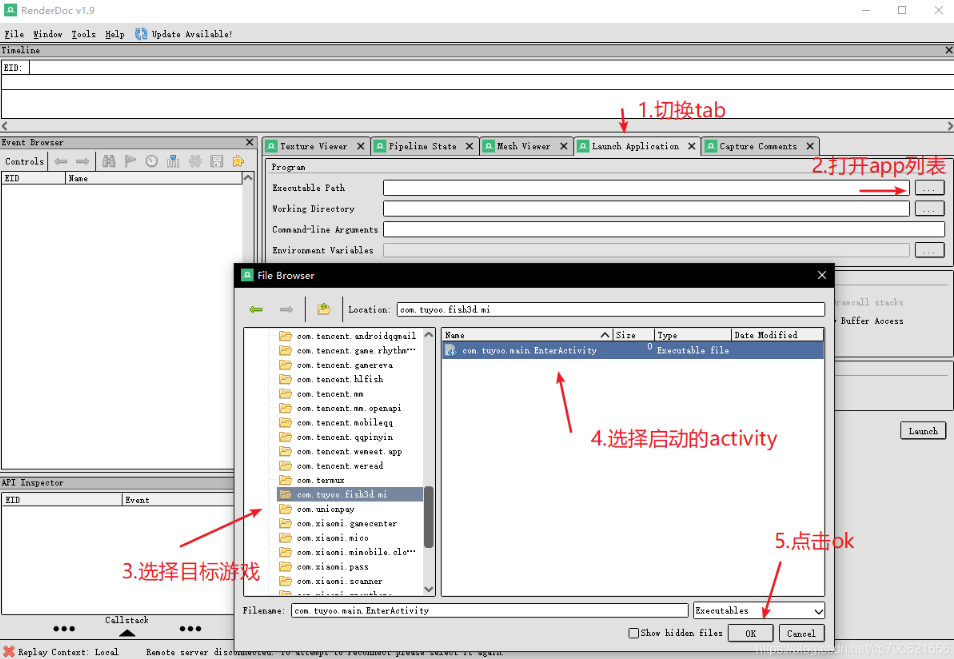


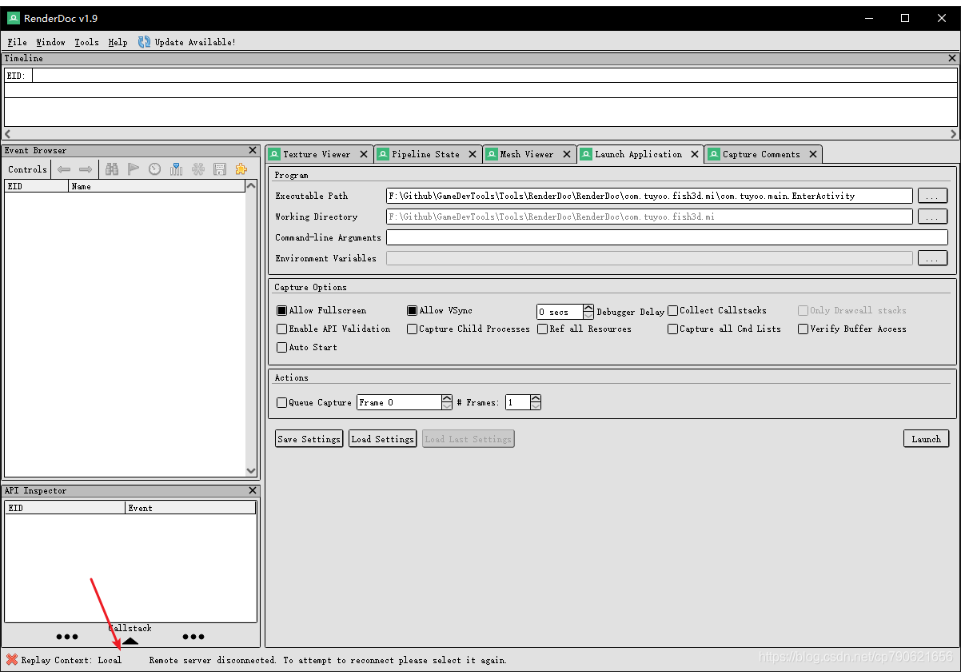
上图表示手机上的RenderDocCmd已经启动成功，可以开始调试了。

同时，你的手机上应该是这样的：



3.2 调试目标游戏

按上图选择目标游戏启动Activity。

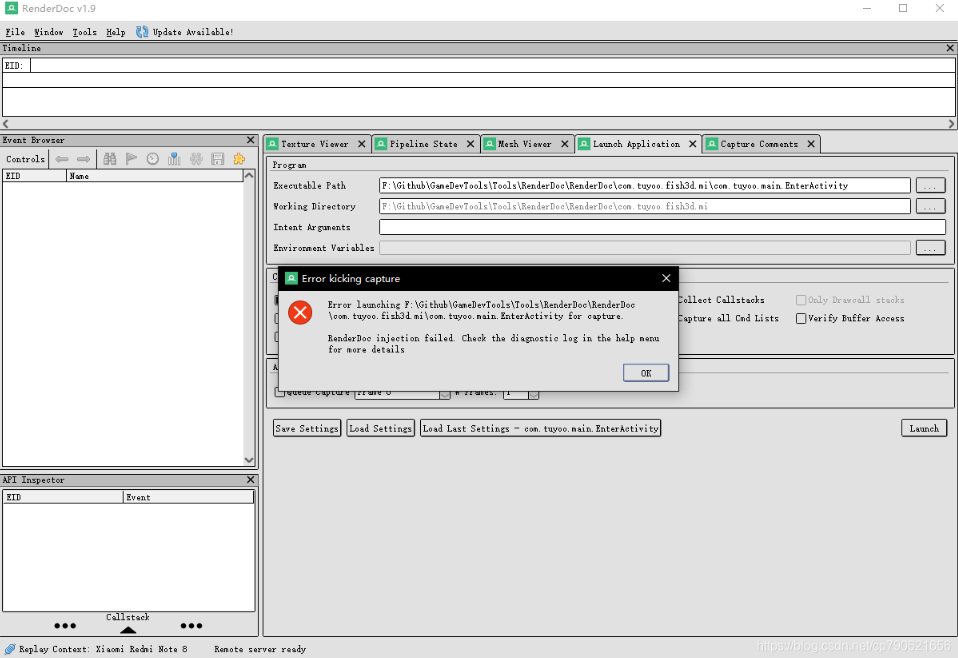


选择好之后，手机会断开连接。

重新选择手机连接。

点击Launch按钮，开始调试目标游戏。

一般都会收到下图错误。



这是因为，目标游戏是打的Release模式APK，而不是Debug模式的。

那怎么办呢？

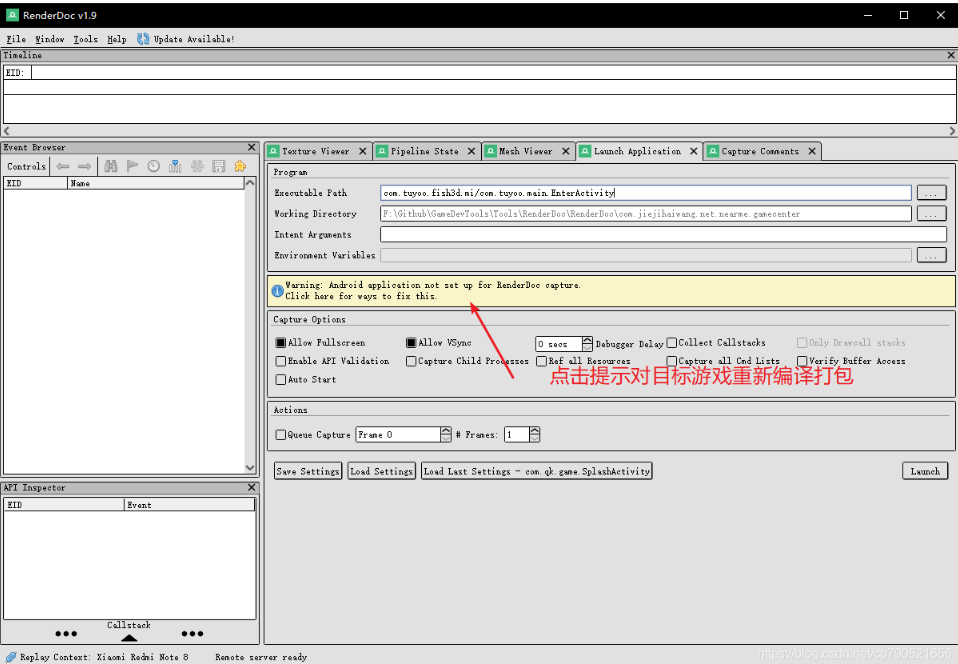
不要急，RenderDoc为我们提供了一键重新打包服务。

来一番骚操作。

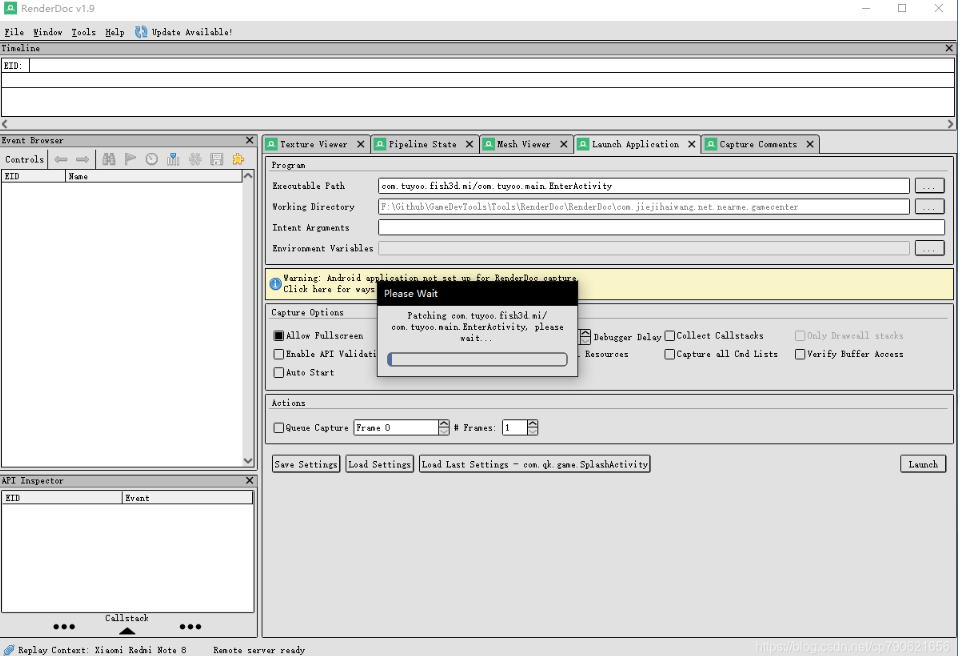
首先，选择另一个游戏，然后Launch，应该也是上面的错误。

然后再选择目标游戏。

这样来回选择，直到选择目标游戏的时候，出现下图的提示。

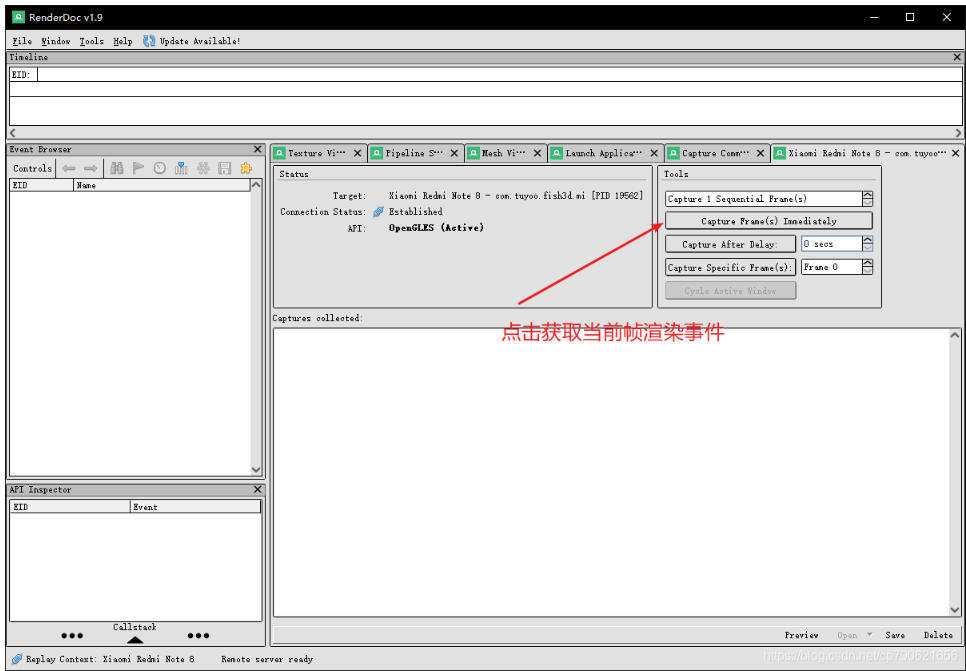


点击黄色区域，就会自动对选中的游戏进行重新打debug包了。

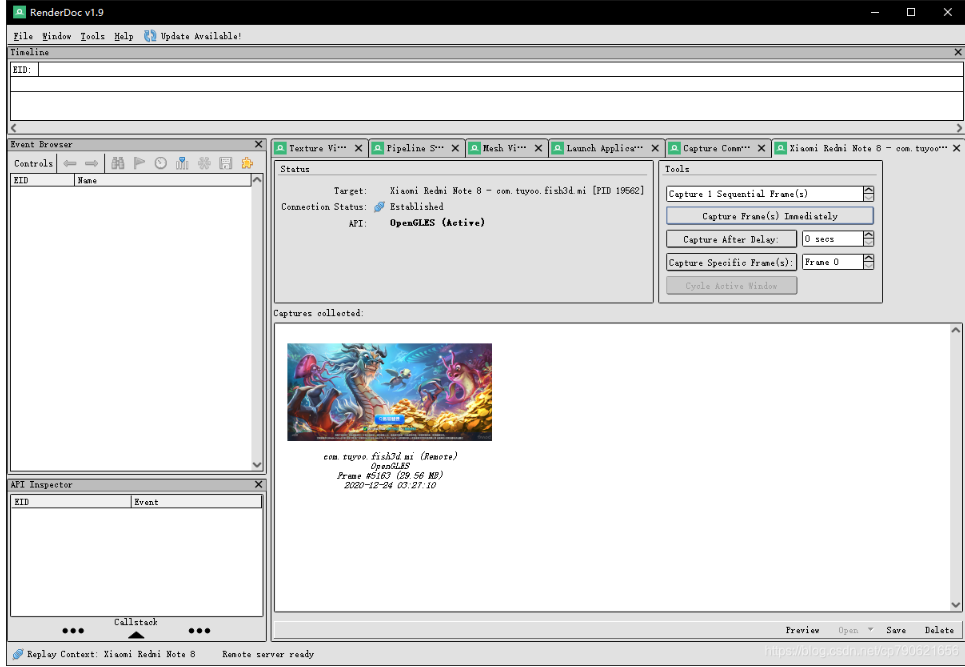


重新打包完成之后，再次点击 Launch，这次一定成功，目标游戏在手机上自动运行起来了。

RenderDoc界面也发生了变化。



点击箭头所指按钮，即可截取当前帧所有渲染事件。



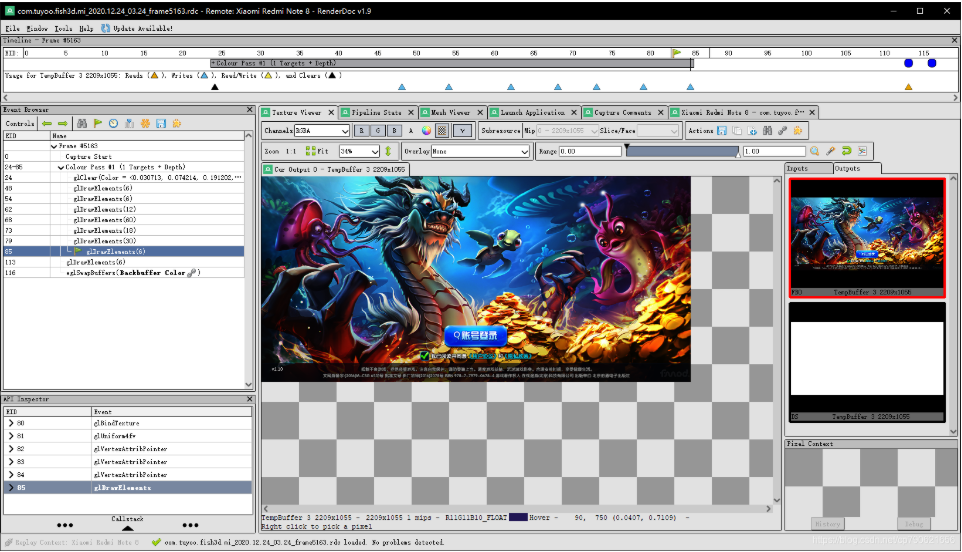
截取的这一帧数据，显示在列表区域中。

此时截取的这一帧数据，仍然保存在手机上，一旦手机断开，就不可用。

所以最好立即保存到电脑磁盘中。 对封面图片右键，选择

save，然后找个地方存起来。

双击封面图片，打开详细Drawcall列表。



左侧列表中，就是每一次Drawcall。

中间的大图默认就是当前Drawcall后的效果。

右边的小图是输入输出。本文转自https://huutu.blog.csdn.net/article/details/111659729

和Unity的FrameDebuger类似，其实所有的GPU分析工具都差不多。

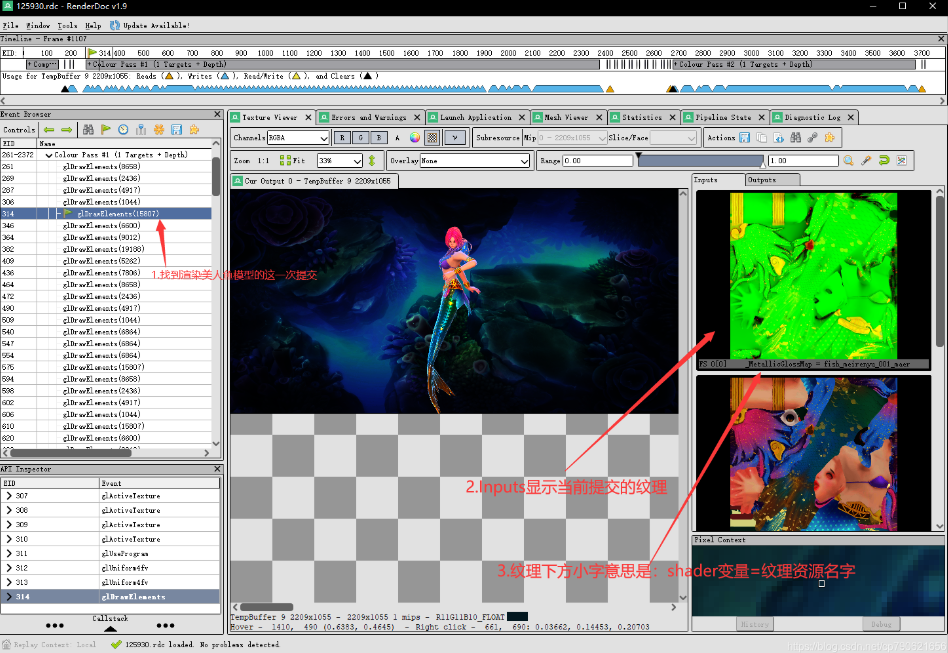
4.RenderDoc分析具体场景

下面使用RenderDoc来分析开始提到的3个问题。

4.1 美人鱼的三张纹理中，绿色的那张是什么用途？

在RenderDoc中开始调试目标游戏，玩一会儿游戏，等到出现美人鱼的时候，截取一帧数据进行分析。

在左侧渲染事件窗口，鼠标单击，找到渲染美人鱼模型的地方。



当前界面示意如上图。

右侧Inputs纹理列表，表示当前提交的纹理资源，以及shader变量名字。

从变量名字，就可以看出这个纹理的作用。

绿色的纹理，传递到shader的MetallicGlossMap变量，说明这是一张金属光泽的贴图。

其他贴图名称及作用

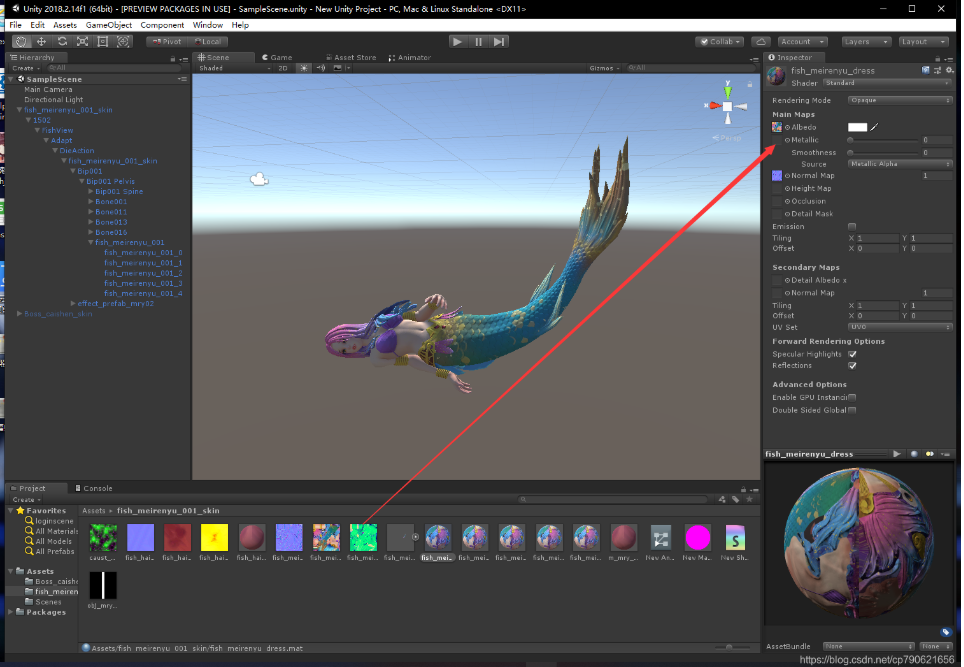
BumpMap 凹凸贴图(用来呈现衣服褶皱之类的凹凸效果

AlphaMask 不透明度贴图(可以让你的服装呈现半透明状态)

MetallicGlossMap 金属反光度贴图(影响质感跟光泽度显示的一张贴图)

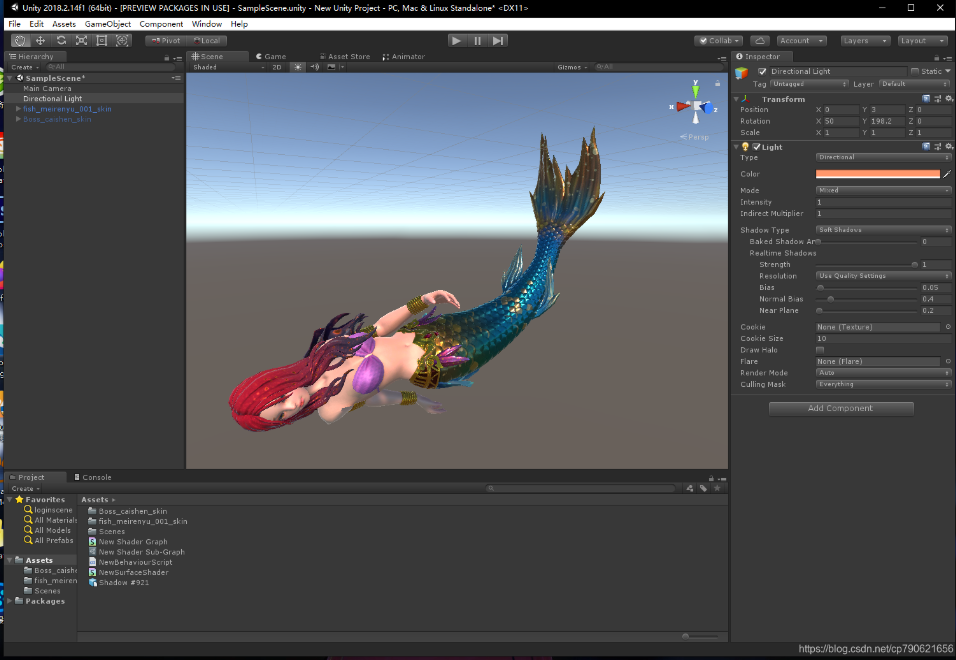
这个名称很熟悉，在Unity Standard Shader里，就有。

那么不管目标游戏是不是用Unity Standard Shader，我这里就先用了。



将绿色贴图，拖到 Metallic里。

效果如下:



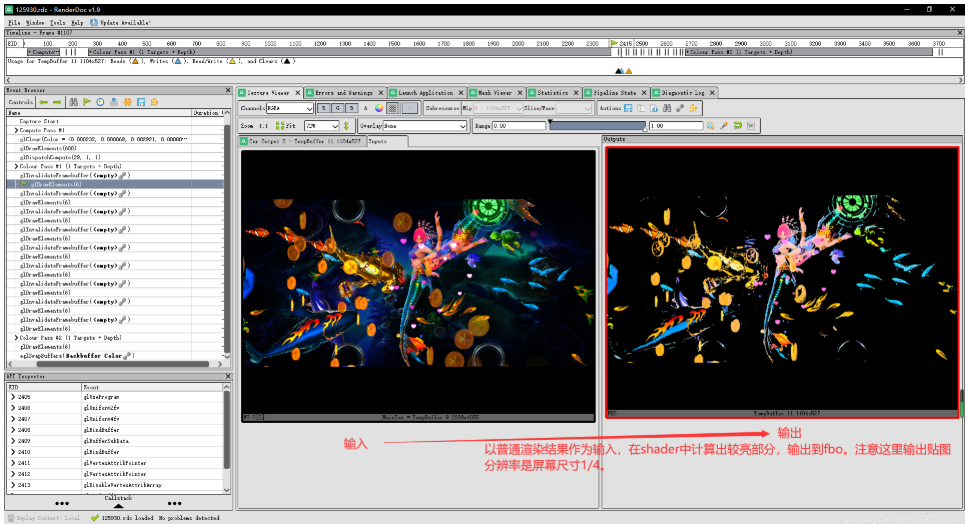
4.2 游戏中bloom的方式

Bloom 俗称全屏泛光，就是物体周围有一层光晕的效果，其实就是近视眼不带眼睛看灯的感觉。

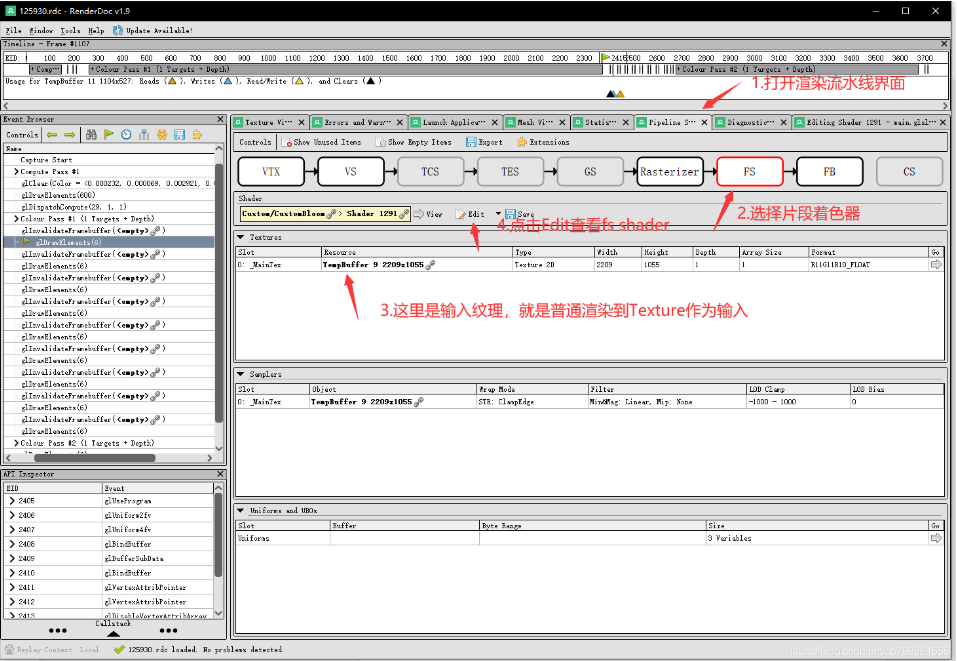
简单介绍Bloom就是，先普通渲染一次，然后渲染亮度高的物体一次并进行若干次高斯模糊，然后将两次渲染结果混合。

对目标游戏截取一帧进行分析。

步骤1：将普通渲染结果输出到Texture，将其作为输入，计算出较亮部分，输出到Texture。

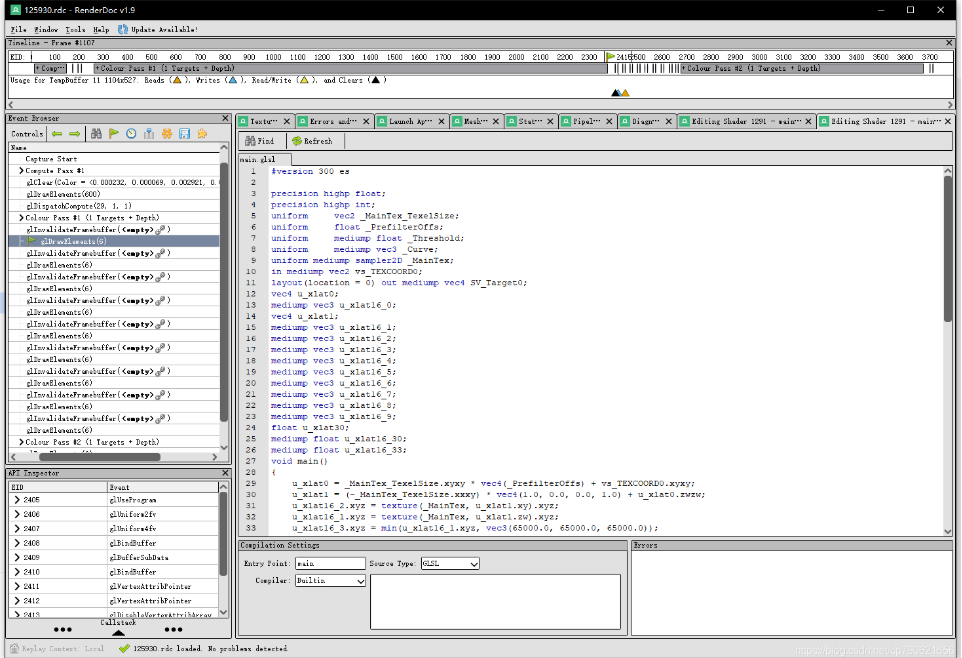


具体是怎么实现的呢，切换到渲染流水线选项卡 Pipeline State。



后期特效都是在FS中完成，所以这里查看fs shader代码即可。

按图中，第4步，点击Edit查看fs shader代码。

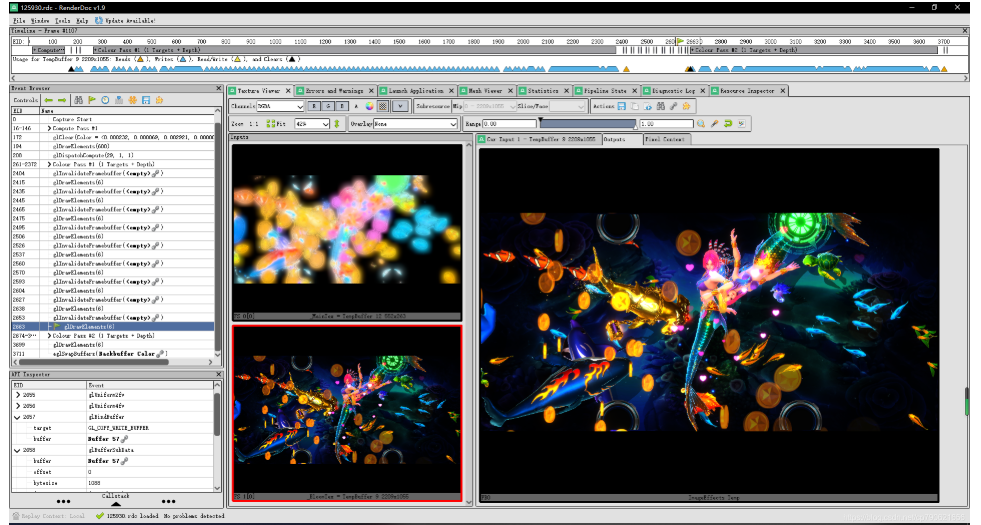


继续按照上面步骤1的方式，对左侧列表渲染事件进行查看。

总结Bloom步骤如下：



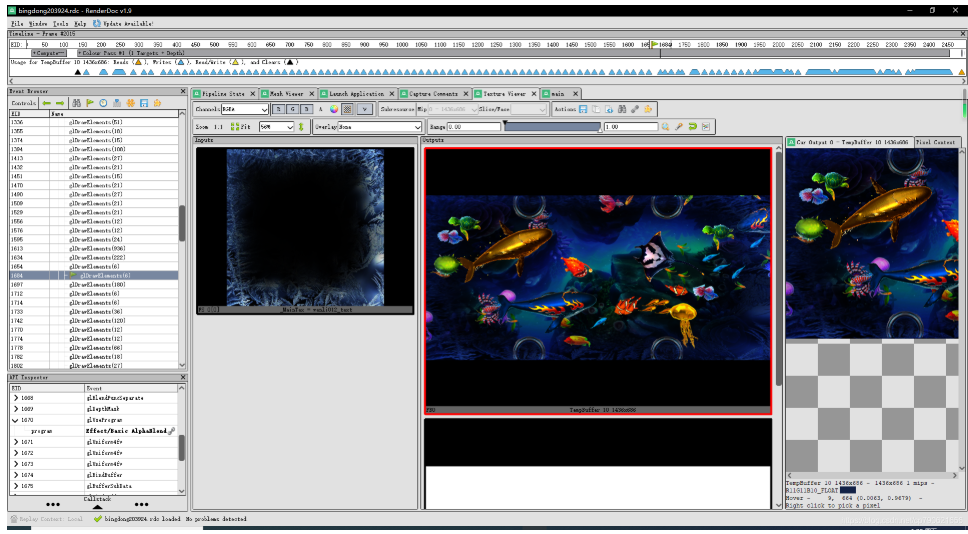
最终效果



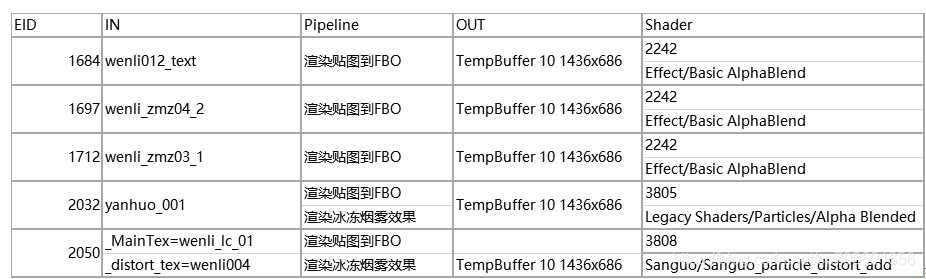
4.3 冰冻特效如何制作

步骤和上面一样，在游戏中出现冰冻效果时，截取一帧。

找到冰冻效果最开始的EID 1684，按顺序分析。



值得注意的是，这里输出到FBO 尺寸比我的手机尺寸小，这是游戏自动降低画质了。

总结步骤如下：

如果要还原效果，就参考Pipeline中给的Shader，在Unity里重写一遍就好。



5.总结

掌握一点OpenGL基础之后，就可以尝试去分析其他游戏的渲染流程，扶着鹰酱过河。RenderDoc又给我们提供了极大的便利，调试其他游戏更加方便，实乃学习必备神器。