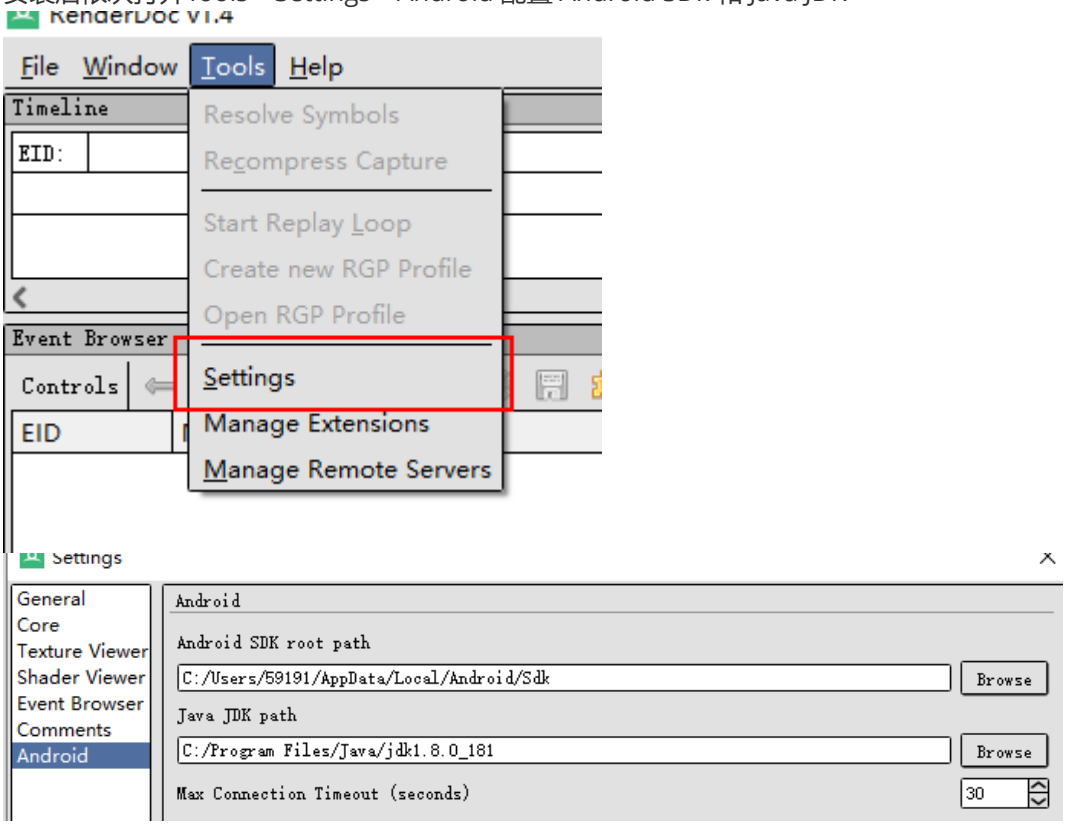
RenderDoc使用

目的

- 优化GPU
- 逆向其他游戏
- 调试Shader

准备工作

- 去官网下载 RenderDoc 官网
  - 安装后依次打开Tools→Settings→Android 配置 Android SDk 和 Java JDK



- 准备一台手机（推荐小米手机）
  - 高通的GPU
  - Android8.x及以下版本
  - Root，并且要解锁System分区。（小米的话Root步骤具体可查看这篇文章）
- 把手机设置为debuggable
  - 下载 mprop
  - 使用命令（需要安装adb，RenderDoc里面也有，目录在 `C:\Program Files\RenderDoc\plugins\android`）

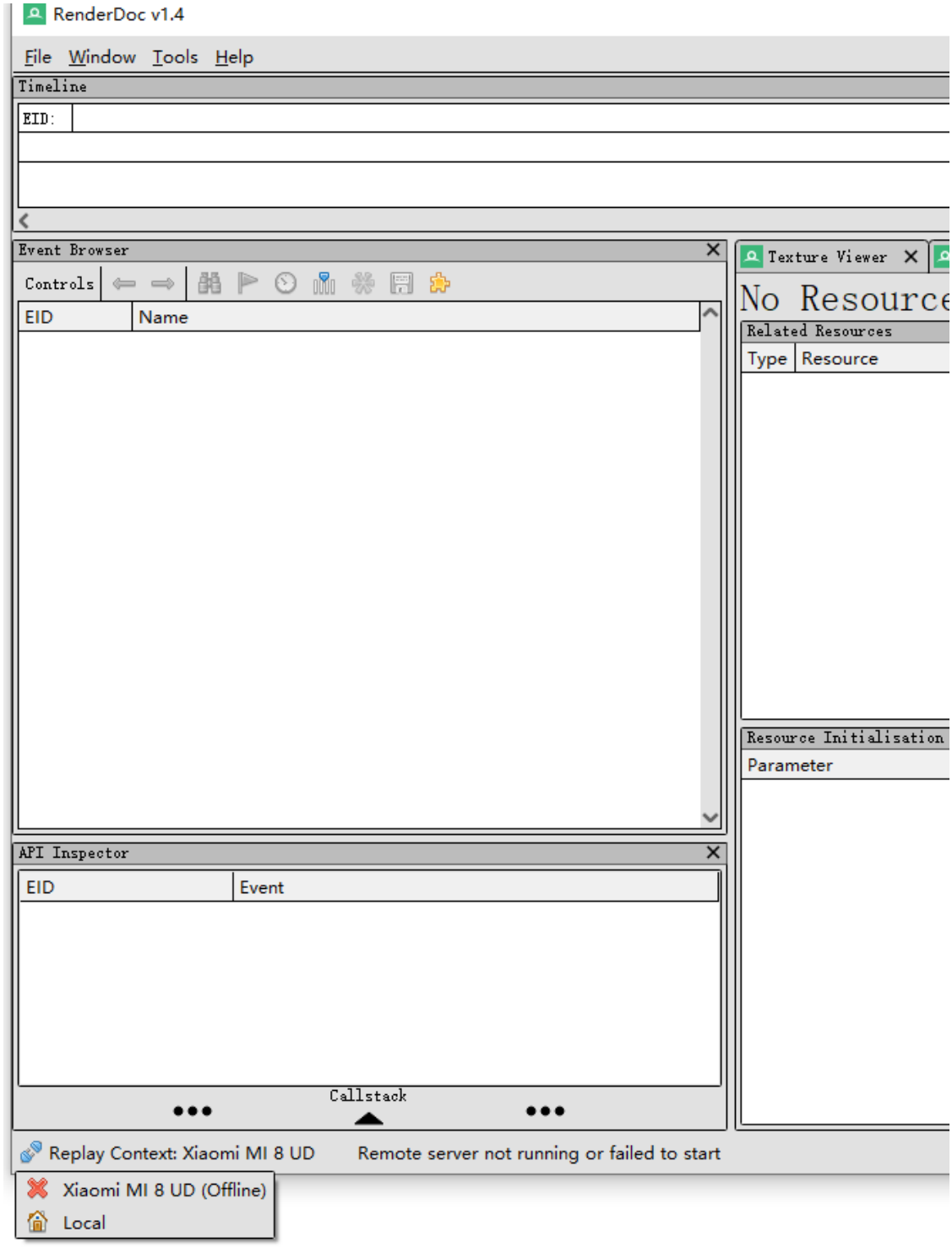
    ```
    adb push mprop-master/armeabi-v7a/mprop /data/local/tmp
    adb shell su
    chmod 755 /data/local/tmp/mprop
    data/local/tmp/mprop debuggable 1
    stop;start
    ```

  - 重启之后使用 `getprop ro.debuggable` 看看输出是否是1，如果是1表示设置成功。

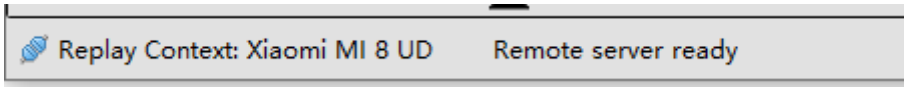
逆向

连接应用

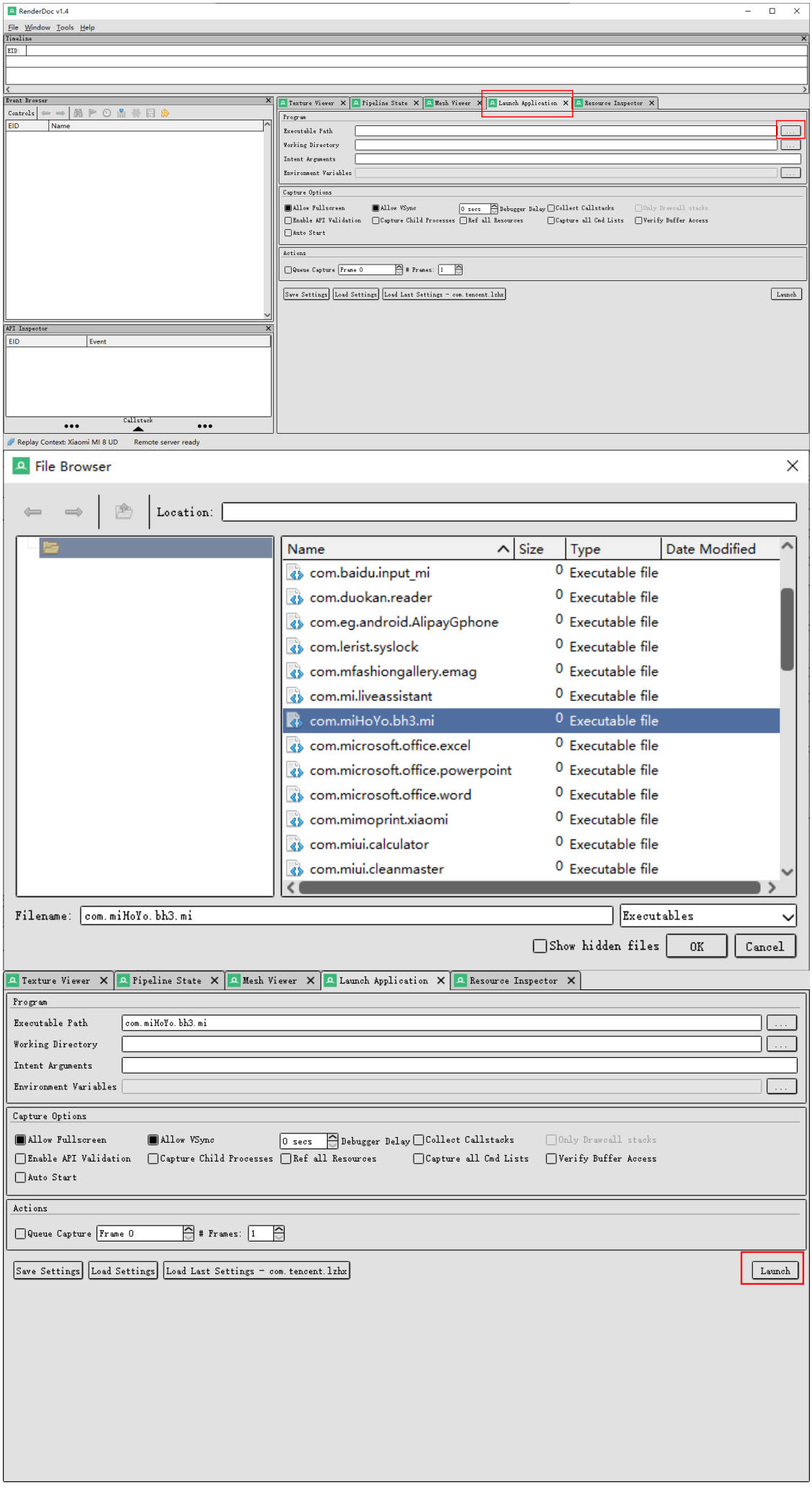
- 打开RenderDoc，点击左下角选择连接的手机，会在手机上自动安装RenderDoc的Android App，保持手机常亮。



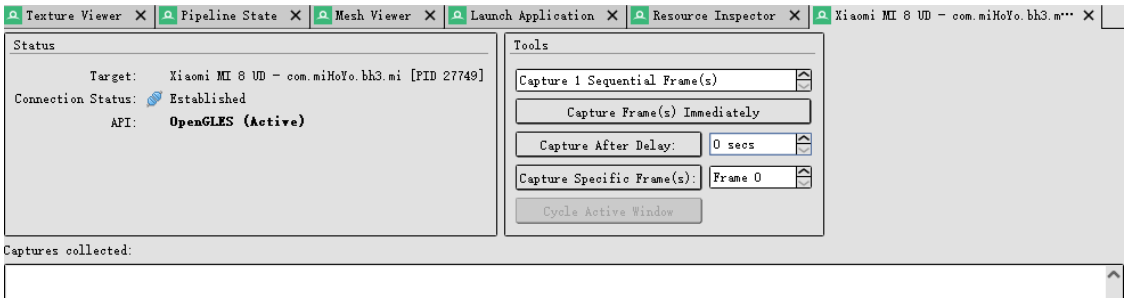
成功连接后会是如图显示



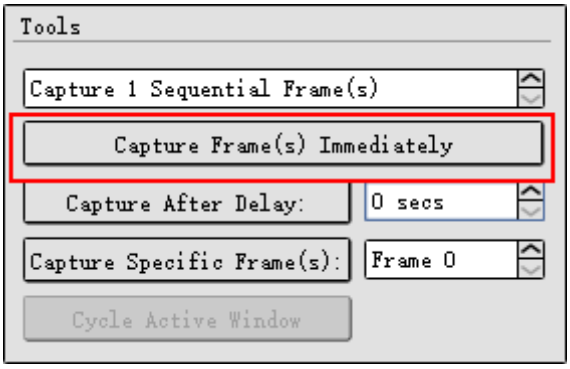
- 选择Launch Application窗口，选择Executable Path，在弹出的窗口中选择你要连接的包名（这里以bh3为例，一般来说可以在手机的设置→应用中看到app的具体包名），ok之后点击Launch按钮


RenderDoc v1.4
File Window Tools Help
Timeline
Event Browser
Texture Viewer
Pipeline State
Mesh Viewer
Launch Application
Resource Inspector
Program
Executable Path
Working Directory
Intent Arguments
Environment Variables
Capture Options
Allow Fullscreen
Allow VSync
Debugger Delay
Collect Callstacks
Only Drawcall stacks
Enable API Validation
Capture Child Processes
Ref all Resources
Capture all Cmd Lists
Verify Buffer Access
Auto Start
Actions
Queue Capture
Frame 0
Frames:
Save Settings
Load Settings
Load Last Settings - com.tencent.lzhx
Launch
API Inspector
EID
Event
Callstack
Replay Context: Xiaomi MI 8 UD
Remote server ready
File Browser
Location:
Name
Size
Type
Date Modified
com.baidu.input_mi
com.duokan.reader
com.eg.android.AlipayGphone
com.lerist.syslock
com.mfashiongallery.emag
com.mi.liveassistant
com.miHoYo.bh3.mi
com.microsoft.office.excel
com.microsoft.office.powerpoint
com.microsoft.office.word
com.mimoprint.xiaomi
com.miui.calculator
com.miui.cleanmaster
0 Executable file
Filename:
com.miHoYo.bh3.mi
Executables
Show hidden files
OK
Cancel

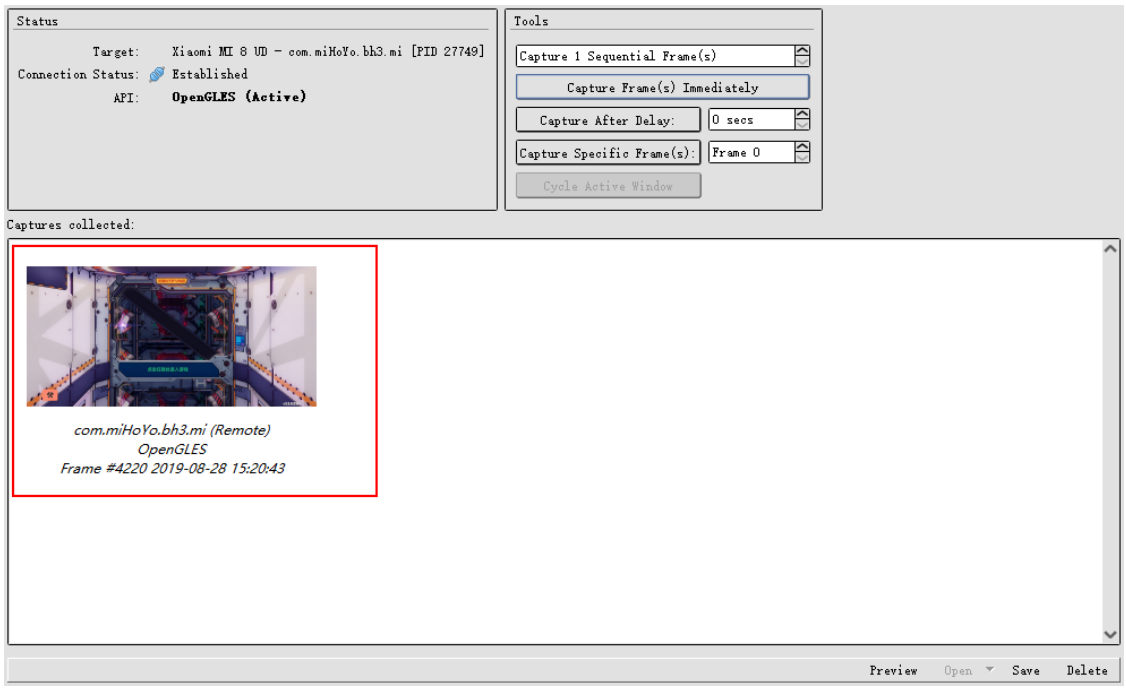
Launch成功后显示如图



截帧

- 点击Capture Frame 截帧 

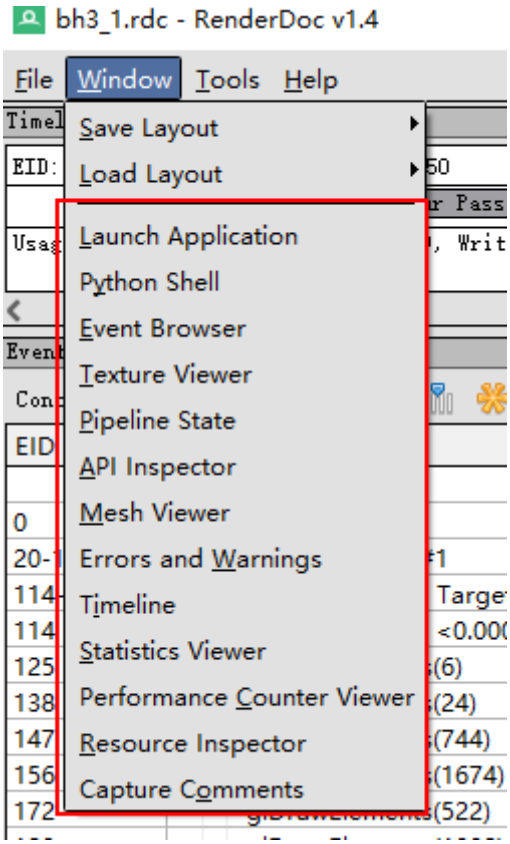
成功之后双击截出的帧，等待加载完数据，就可以开始分析了。当然你也可以右键保存到本地，这样方便下次分析。



分析帧

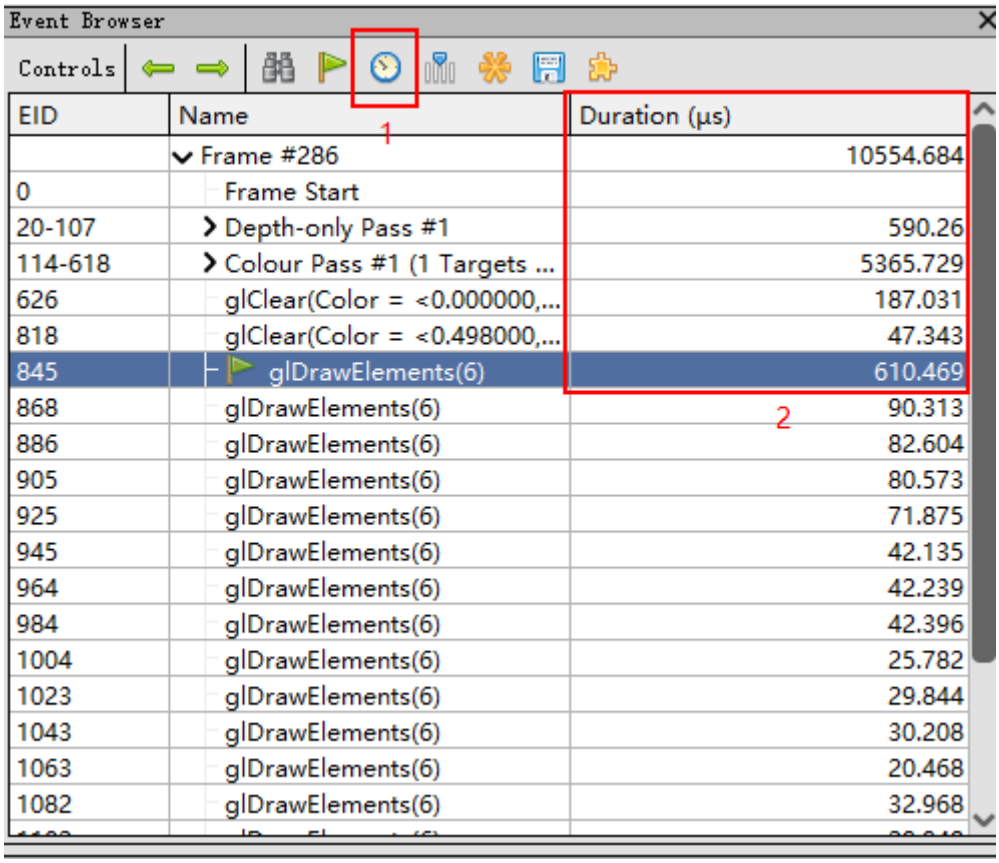
窗口简介

RenderDoc中全部窗口如下：



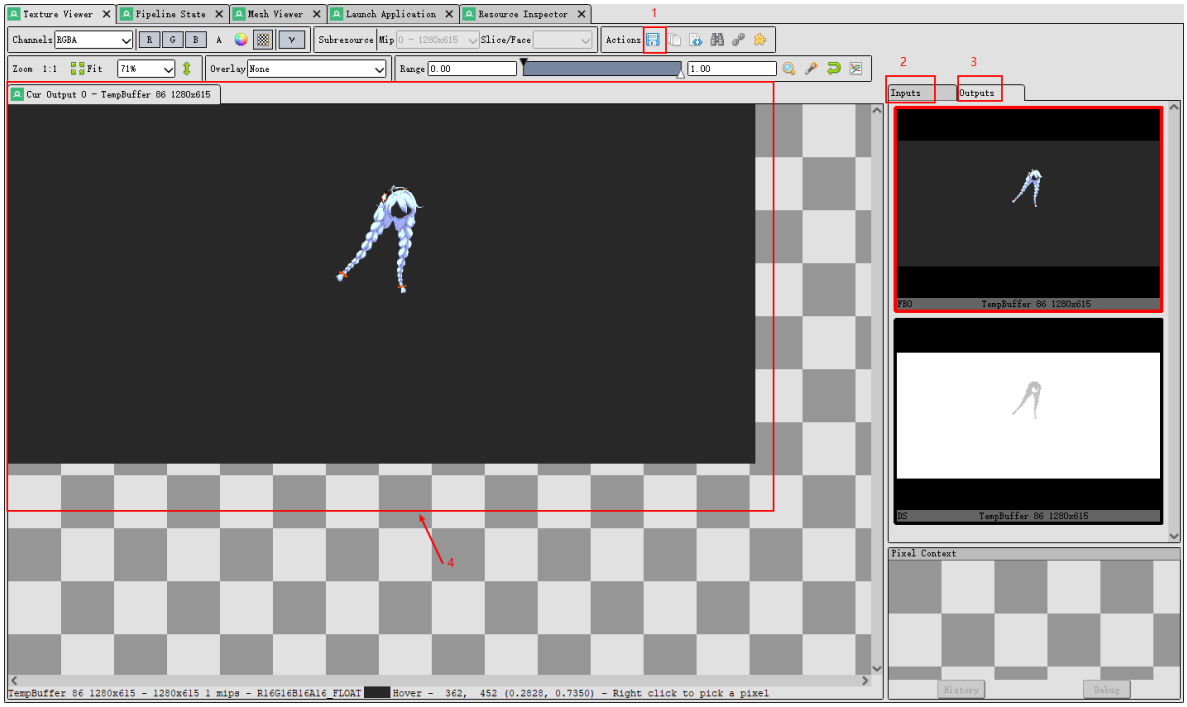
Launch Application上面说过是用来连接应用和抓帧的 Python Shell，Errors And Warnings 我基本不用，无视。 其他窗口下面我会按照我常用频率来介绍。

Event Browaser

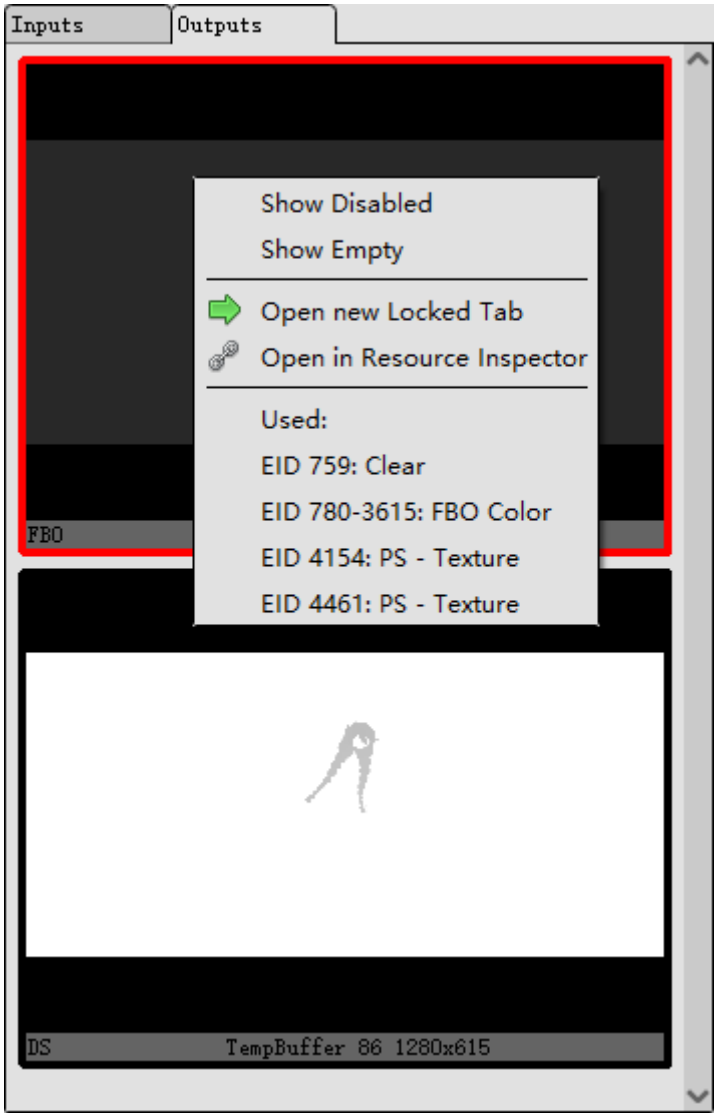


主要用来查看以及选择DrawCall调用，我们分析的单位也是从DrawCall分析。选中单一的DrawCall，其他窗口也会有对应的改变显示，因此在glDrawElements之间跳动，可以很方便的查看到这次Drawcall具体作用。同时点击 1 处的时钟图标，会在 2 处显示调用的消耗时间。

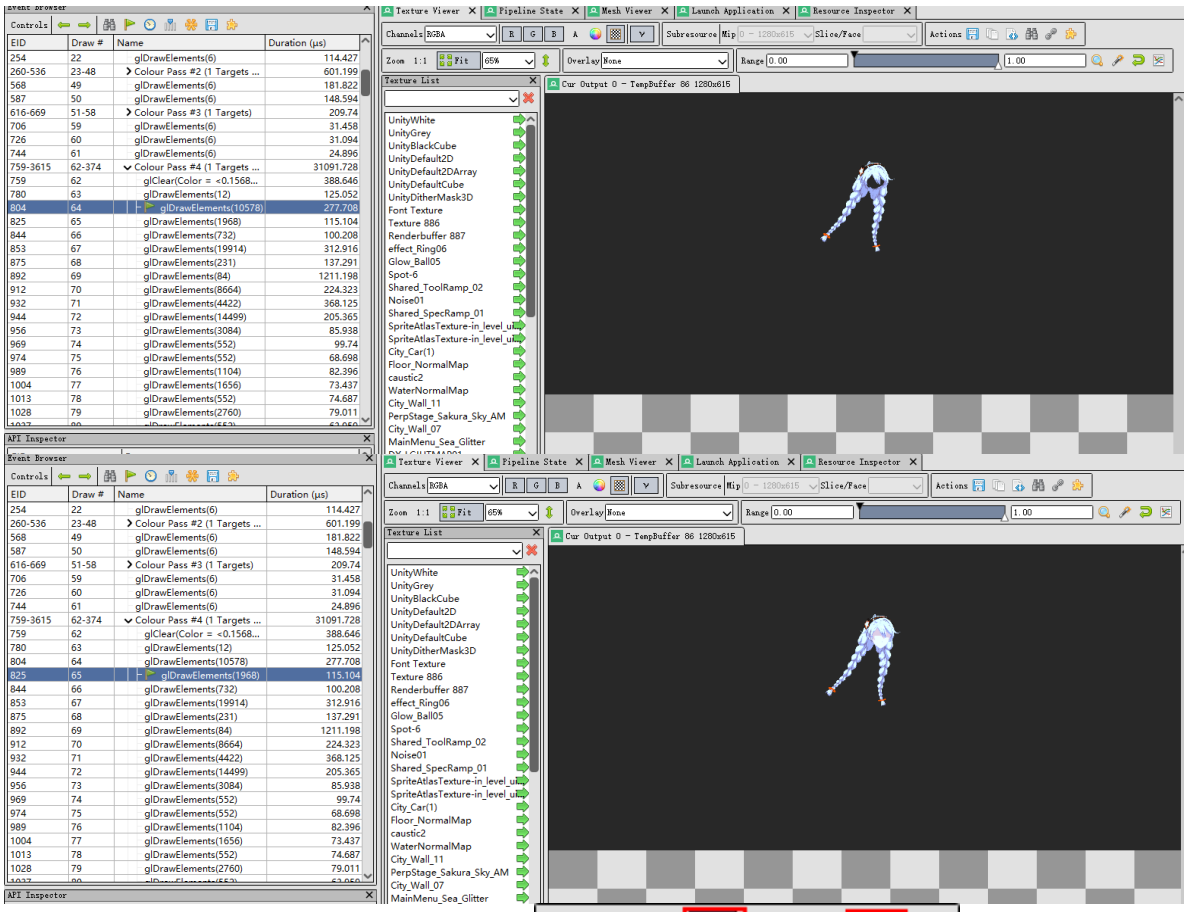
Texture Viewer



主要用来查看纹理显示，我们可以用它来查看在Event Browaser中选中的DrawCall或者Pass的输入和输出纹理。 2 中是输入纹理， 3 中是输出纹理，在其中选中一个纹理，可以在 4 中放大显示。点击 1 按钮可以保存纹理。



在纹理小窗口中右键弹出一些选项，可以看到这个纹理在哪些DrawCall中使用，点击可以Event Browaser中选中对应的DrawCall。 例如我在一次截帧中顺序选中两个DrawCall。



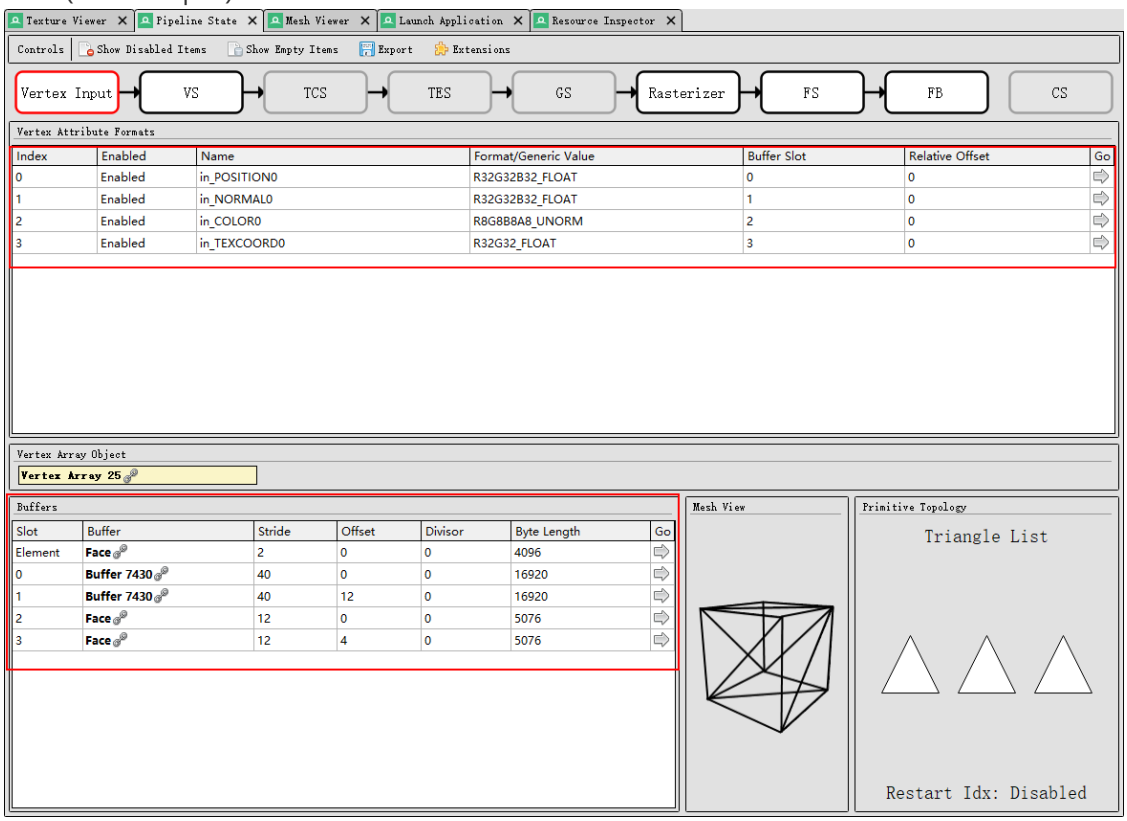
对比发现第二个DrawCall画的是人物的脸部。 

窗口还有两个比较有用的小功能，一个是查看所有纹理，一个是在Resource Inspector中查看当前选中纹理，这里就不多做介绍。

Pipeline State

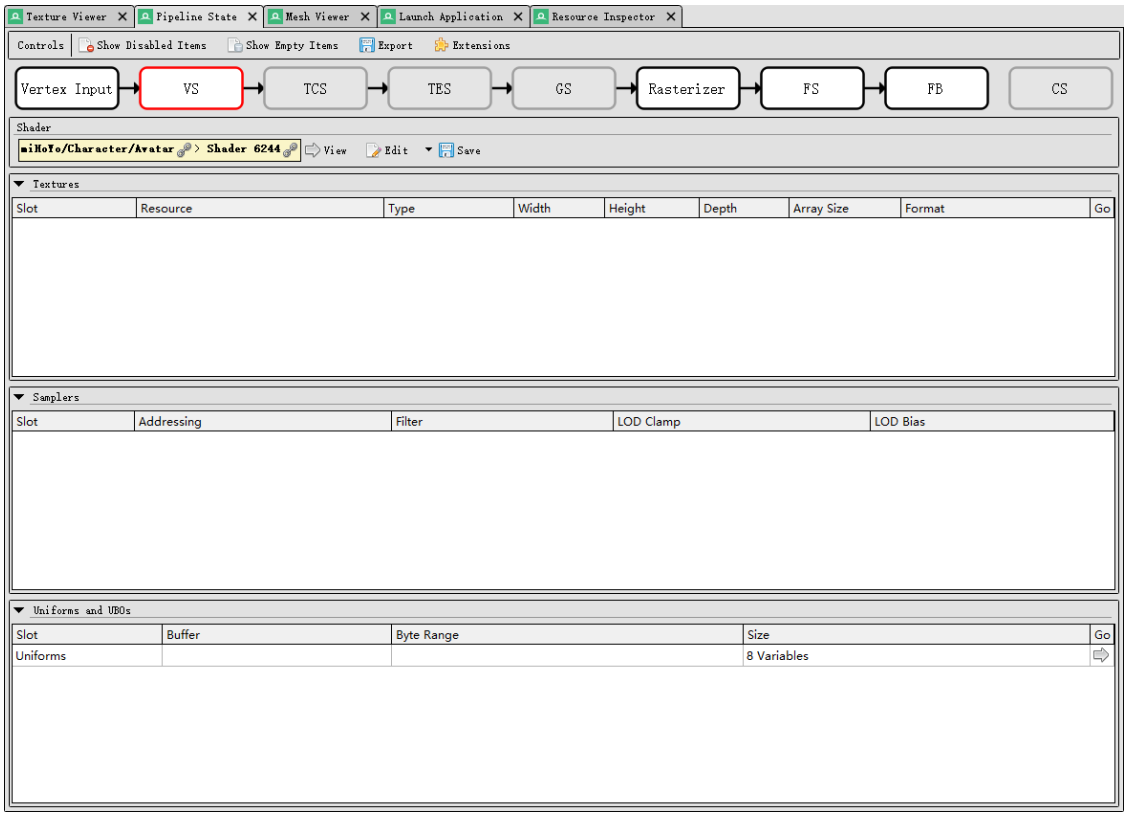
主要用来查看渲染管线过程，是一个使用很频繁的窗口。(下面介绍以我们绘制脸部的DrawCall为例。)

- VTX（Vertex Input）



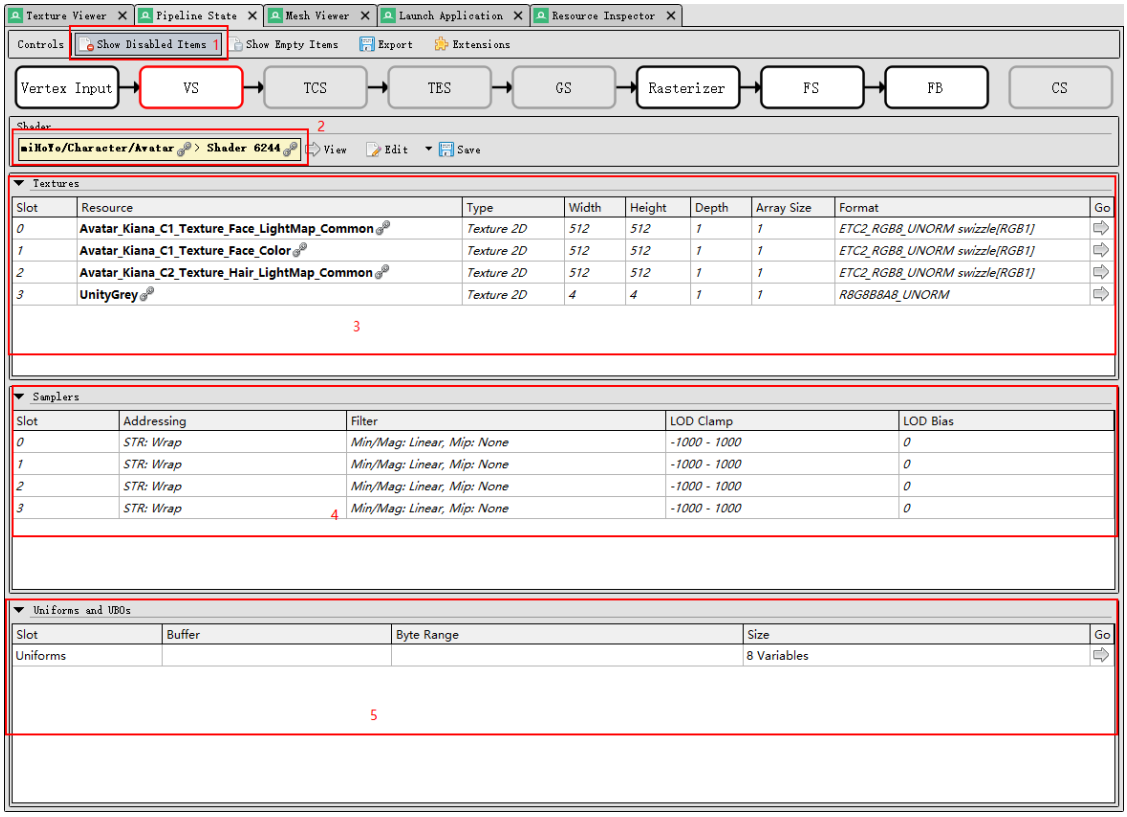
VTX窗口主要用来查看输入的顶点（包括格式，大小等等），粗略看看就行，后面有更适合看的地方。

- VS（Vertex Shader） 这个就是顶点着色器了，如图。

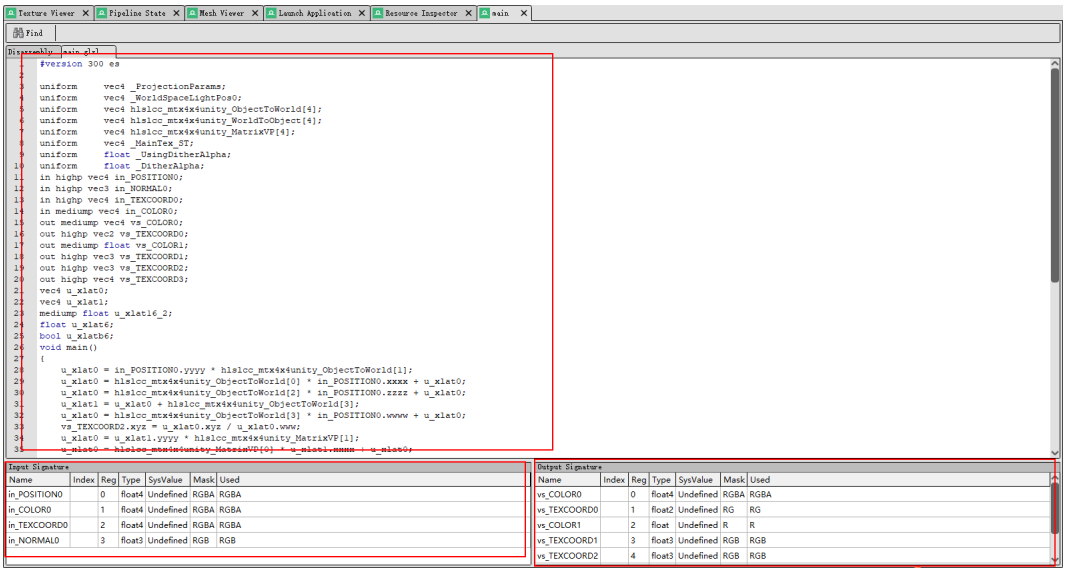


因为基本的上不在顶点着色器中做采样操作，所以这里的纹理和采样器是空的，但是因为后面的PS（像素着色器）和这里基本上是一样的显示和操作，我就点击`Show Disabled Items`（下图中`1`处按钮）强行显示出来这个ShaderProgram使用到的纹理和采样器（一般情况下不需要开这

个），PS处就不再细介绍了。



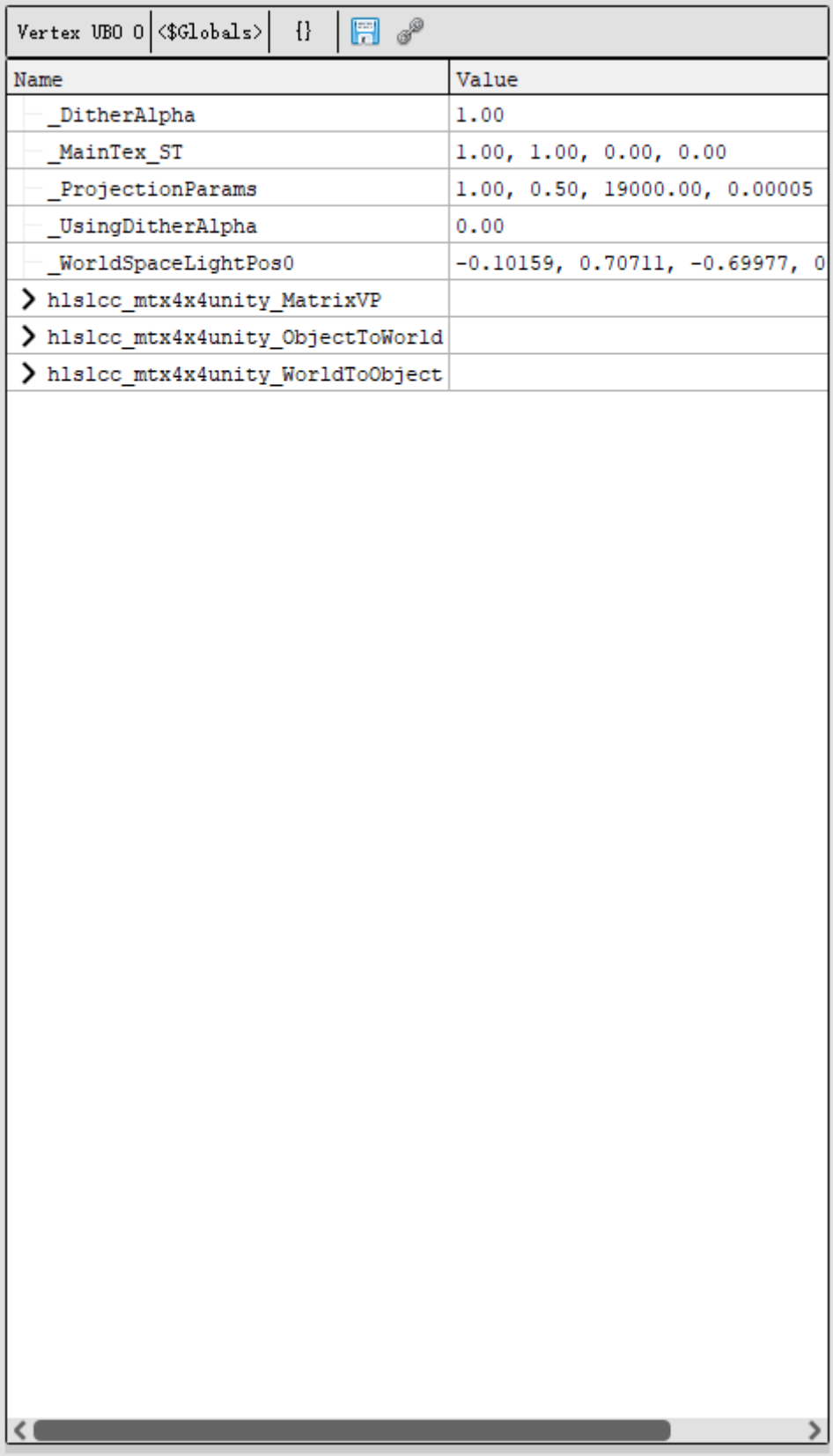
- 2处显示的是ShaderProgramma，可以看到Shader的名字，点击会调到Resource界面，可以看到更详细的内容，后面再介绍。点击`View`按钮可以看到具体的shader代码，如图。



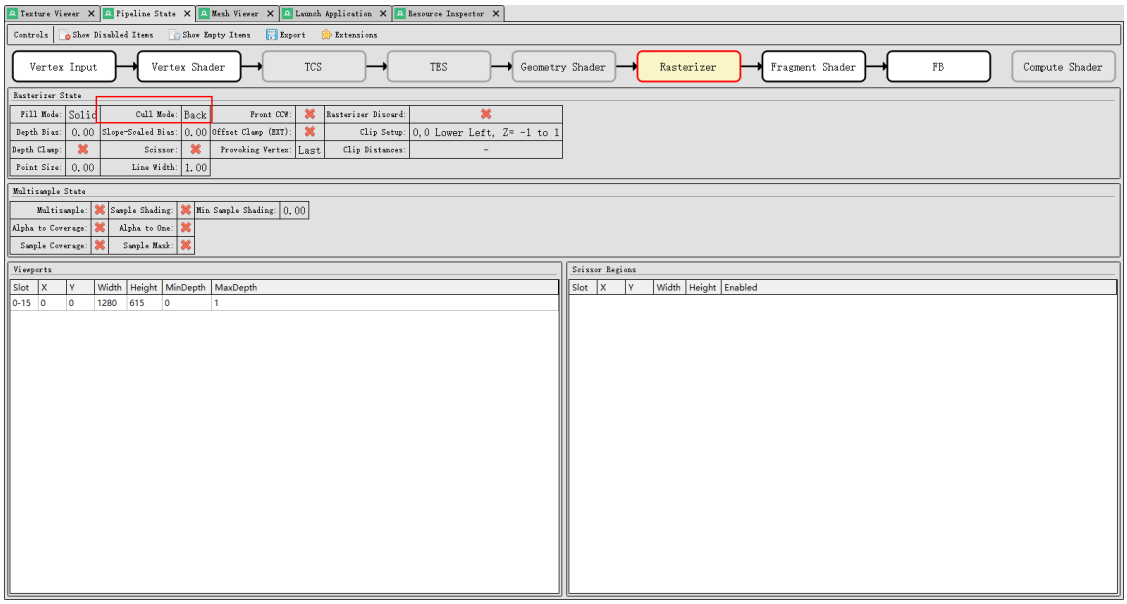
分别是VShader的代码和VS的输入和输出。在Mesh View 能看到更详细的Mesh数据，这个后面再介绍。

- 3处是VS用到的纹理，可以看到纹理类型，大小，格式等。点击➡️可以在Texture View看到纹理显示，点击🔗可以在Resource Inspector中查看。一般来说➡️按钮表示有更详细的信息，🔗按钮表示在Resource Inspector有更多的信息。
- 4处是采样器，可以看到采样纹理的环绕方式（Addressing），过滤方式（filter）等。

- 5 处是VS用到的一些Uniform变量，点击➡可以看到具体的变量和内容。

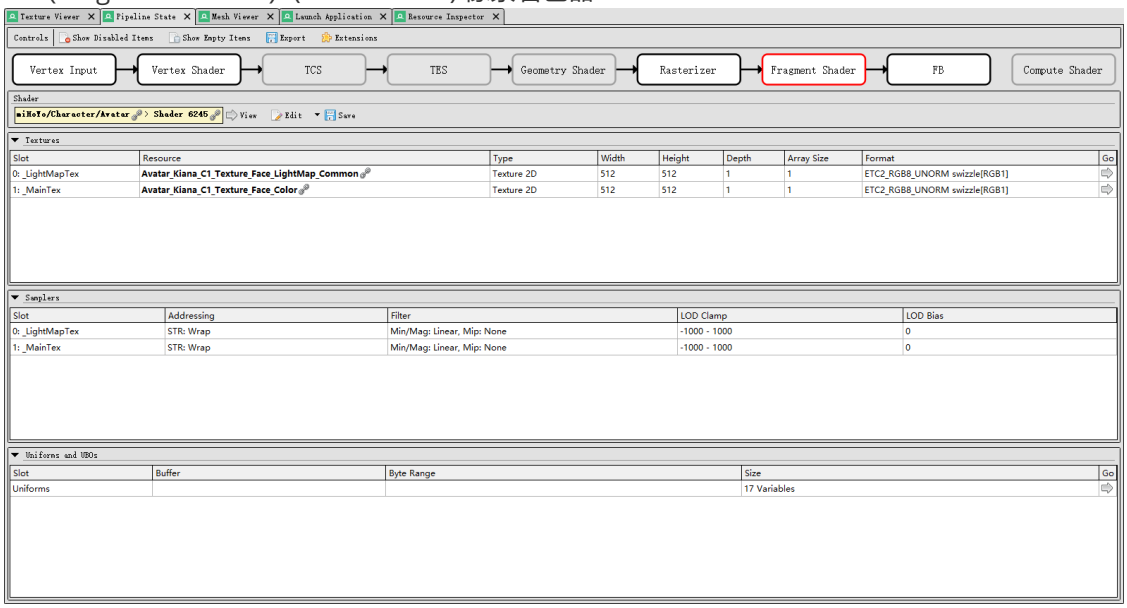


- RS（Rasterizer State）



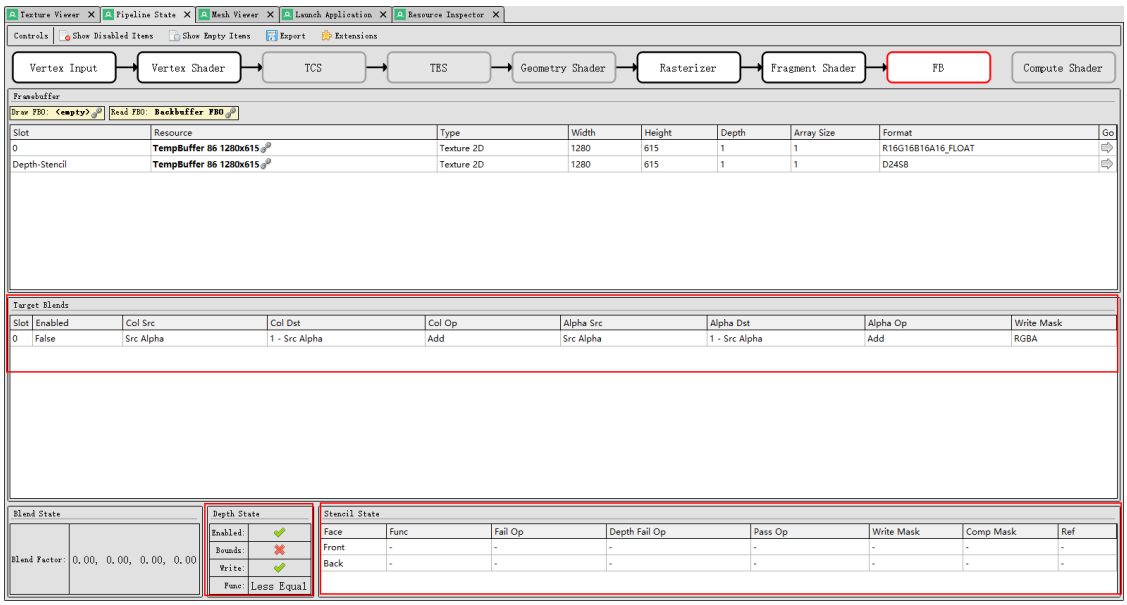
这个窗口我基本只关注一个 CullMode（裁剪模式），其他变量有兴趣的自己去了解哈。

- FS（Fragment Shader）(Pixel Shader) 像素着色器



操作显示和VS差不多，稍微注意下我没开`Show Disabled Items`，因此我们在上面看到的纹理和采样器就是在PS中真正用到的。

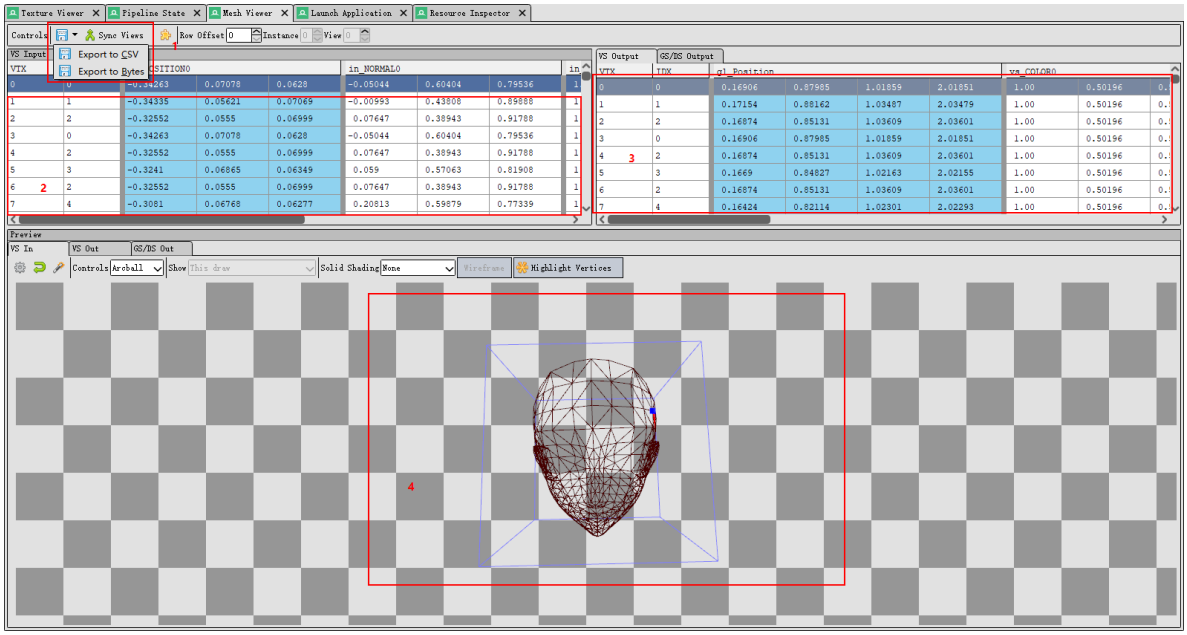
- FB（FrameBuffer）



这个窗口主要关注Alpha混合（是否开启，以及混合模式等），Z-Test（深度测试是否开启，测试函数等等），Stencil Test（模板测试）

Mesh Viewer

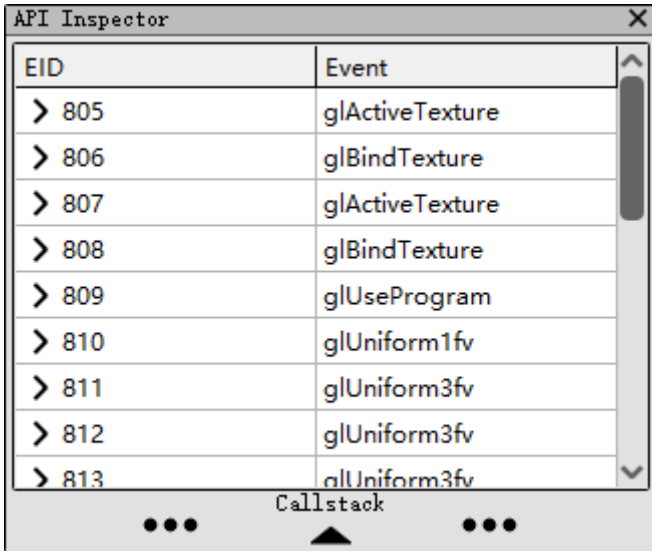
这个窗口主要是用来查看VertexShader中输入输出的顶点数据，一般关注输出的就行。



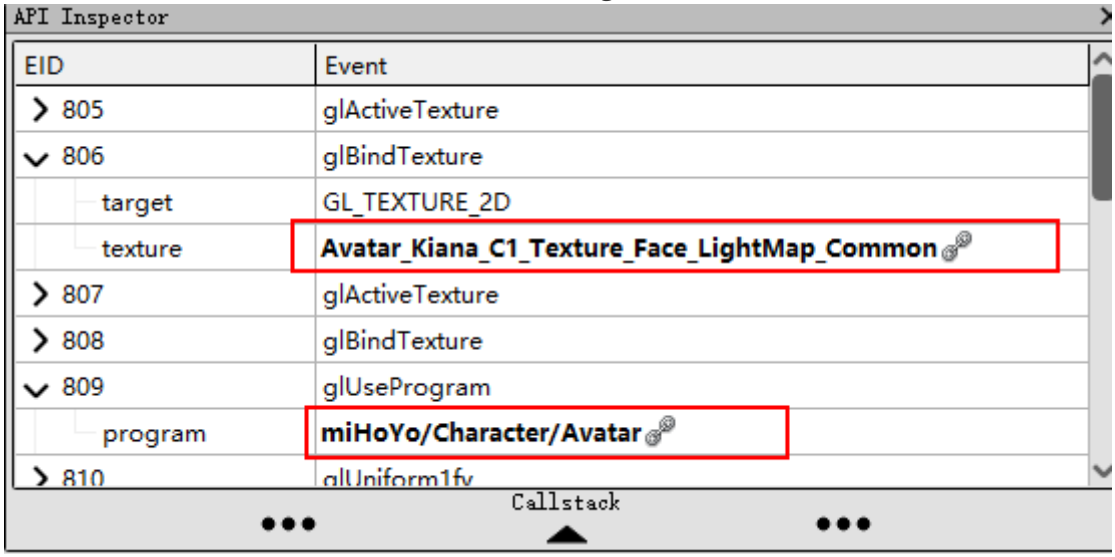
- 2 中的就是输入的顶点数据，3 中是输出的顶点数据，4是你输入的Mesh预览，在 2 中选中任意一个顶点在 4 中都会以蓝色高亮显示，并且以红色高亮显示当前顶点所在的三角形。
- 在 2 中选中任意以后，点击 1 出按钮，可以导出Mesh数据，一般导出成CSV格式，可以根据顶点数据自己生成模型去做测试。

API Inspector

主要用来查看一些切换渲染状态的API调用。



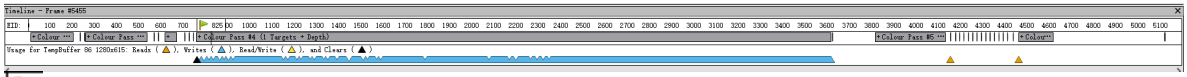
上图可以看到有一些激活绑定纹理，使用ShaderProgram，设置Shader的Uniform变量。



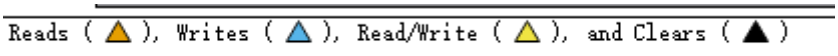
展开后，能看到具体的资源，点击可以跳转到Resource Inspector查看。

Timeline

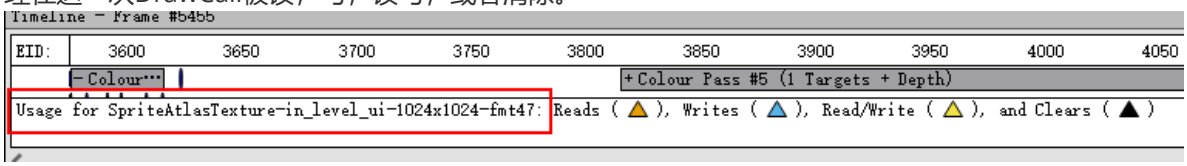
查看这一帧的时间轴



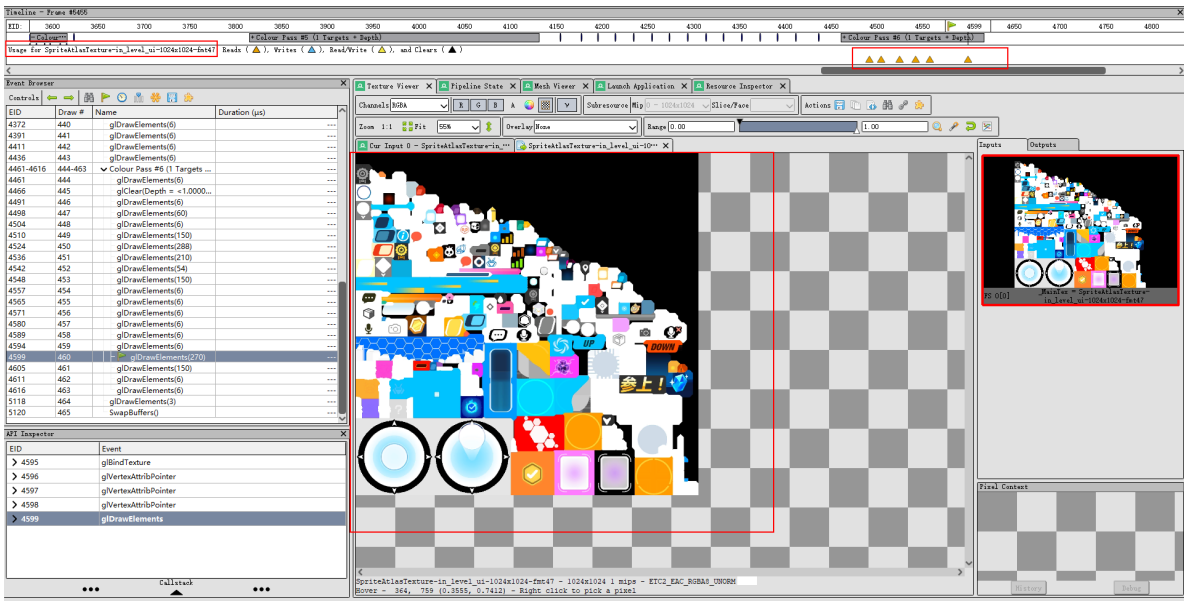
表示当前选中的DrawCall。 当我们在TextureViewer中选中一张纹理，时间轴下面会有一些不同颜色的小三角形。



表示这张纹理在这一次DrawCall被读，写，读写，或者清除。



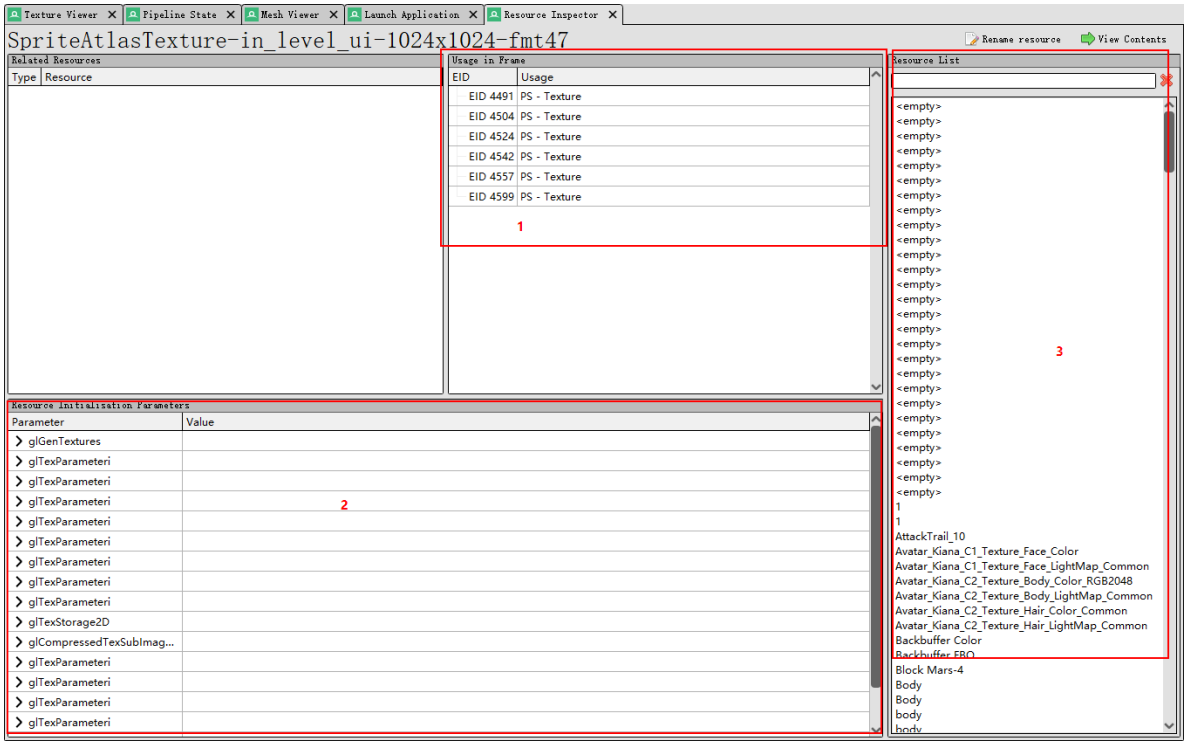
在最左侧可以看到当前查找引用的纹理。 我们选中一张纹理举例。



可以看到这张纹理在6个DrawCall中被读取使用。

Resource Inspector

查看这一帧中所有的资源。



- 在`1`中可以查看到所有引用到这一个资源的的EID，点击可以跳转到Event Brower中选中。
- `2`中是初始化这个资源调用的一些OpenGL API，以及调用API传递的参数。（注意这里只是初始化参数，后面改了的话这里也不会改，例如Shader传了不同的Uniform参数，这里看到的也只是第一次的，所以要看真正的值还是要在在PIpeline State中查看。）
- `3`中是所有的资源列表。