UI优化动静分离实践

Unity 2019.4.3f1

目录

[动静分离 1](#_Toc83387007)

[测试项目结构和结论 1](#_Toc83387008)

[动静分离实践总结 3](#_Toc83387009)

[Part I 不做动静分离 3](#_Toc83387010)

[D20S20 3](#_Toc83387011)

[D20S100 3](#_Toc83387012)

[D100S20 4](#_Toc83387013)

[D100S100 4](#_Toc83387014)

[Part II 做动静分离 5](#_Toc83387015)

[D20S20 5](#_Toc83387016)

[D20S100 5](#_Toc83387017)

[D100S20 6](#_Toc83387018)

[D100S100 6](#_Toc83387019)

[数据分析用途 7](#_Toc83387020)

[UIBattlePad分析数据 7](#_Toc83387021)

[参考资料 10](#_Toc83387022)

# 动静分离

Canvas负责将它包含的几何体合批，生成合适的渲染命令发送给图形系统，当其子物体发生某类变化时，该Canvas会变脏，然后触发rebuild。

动静分离优化UI的原理是：用Canvas来隔离频繁刷新的UI和不刷新的UI，从而减少Canvas rebuild的元素数量，进而起到优化效果的。

更多原理细节：

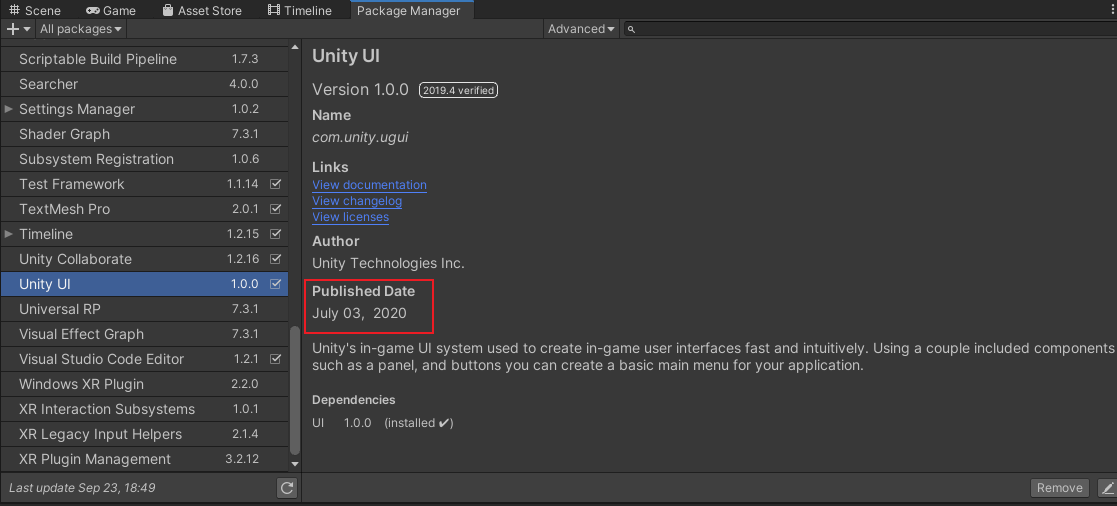
<https://learn.unity.com/tutorial/optimizing-unity-ui#5c7f8528edbc2a002053b5a3http://www.manongjc.com/article/75137.html>

<https://blog.csdn.net/gaojinjingg/article/details/103565840>

# 测试项目结构和结论

测试版本相关：

2019.4.3f1



项目结构如下



Unity Profiler采样数: 300

\_dynamic 存放的是 Image —— Color的A通道逐帧变化的

\_static 存放的是 Image 不随帧数变化而变化

**结论：**

1. 增加Canvas会增加合批数量 -> 增加drawcall
2. 增加静态UI，不会提高Canvas.SendWillRendererCanvases()的开销；

在递增\_static下的image后观察**Time** 得到

1. 增加动态UI，会提高Canvas.SendWillRendererCanvases()的开销；

在递增\_dynamic下的image后观察**Time** 得到

1. 动静分离后，Canvas.SendWillRendererCanvases()的时间开销会增加0.02ms 左右的，同时会有开启 **CanvasRenderOverlays**，其时间开销较大
2. 对比观察PlayerLoop，发现**动静分离实际上可能会增大了时间开销**

**Unity Profiler相关函数**

Canvas.BuildBatch——Canvas是dirty后的build batch过程

Canvas.SendWillRendererCanvases——C# 脚本对于Canvas组件的willRenderCanvas事件的订阅的调用

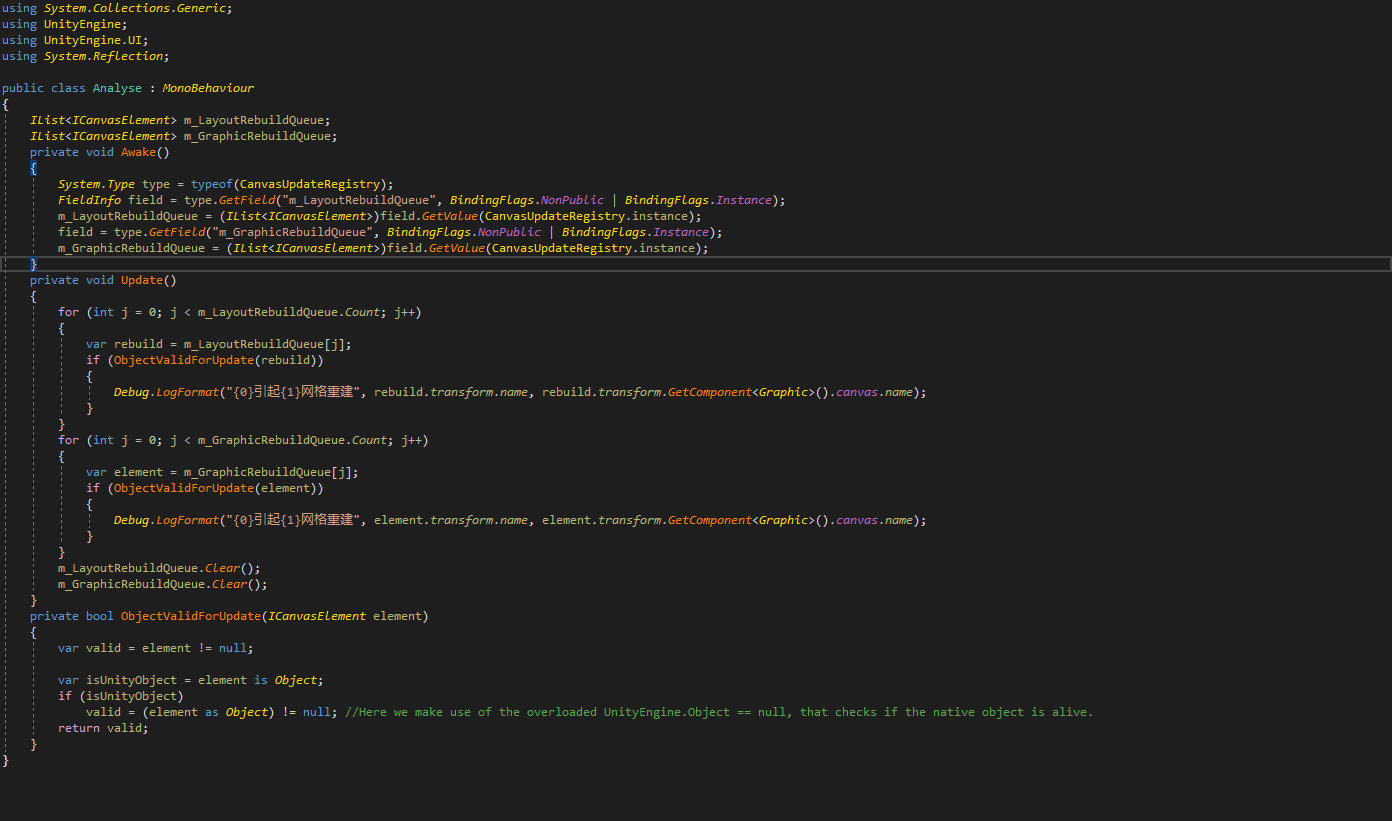
UGUI的drawcall产生的位置取决于Canvas组件上被设置的渲染模式：

1.Screen Space – Overlay将出现在Canvas.RenderOverlays组中。

2.Screen Space – Camera将出现在Render.TransparentGeometry子项，所选渲染相机的Camera.Render组中。

3.World Space将出现在Render.TransparentGeometry子项，每个可以看见Canvas的World Space的摄像机中。

下方代码用于检测哪儿UI元素变化引起了哪儿个Canvas rebatch



# 动静分离实践总结

\_dynamic : Image数： 100 \_static : Image数： 100

|  |  |  |  |
| --- | --- | --- | --- |
|  | PlayerLoop | Canvas.SendWillRendererCanvases | Canvas.BuildBatch |
| 动静分离 | 5.51ms | 1.28ms | 可忽略不计 |
| 不做动静分离 | 1.83ms | 1.24ms | 可忽略不计 |

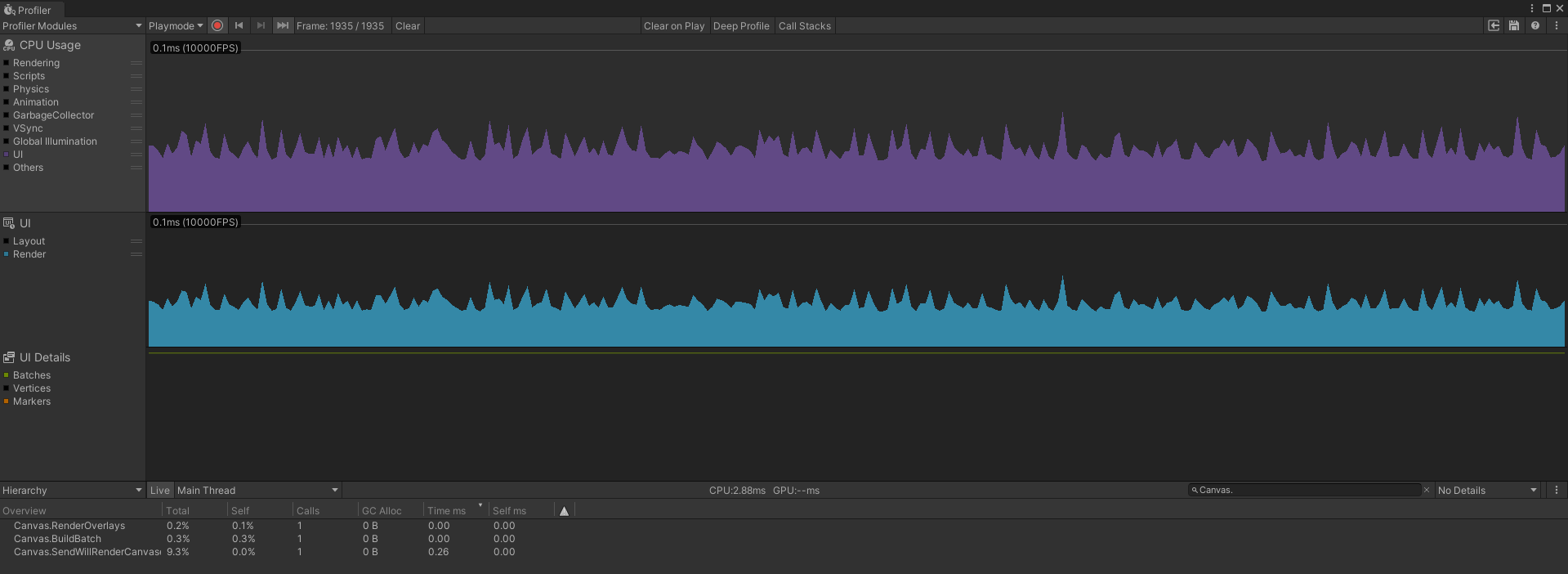
动静分离相关的优化UI的经验和网上资料都是针对unity 5.x.x版本的，与本次实践结果不同。而在unity2019版本,由于动静分离需要增加Canvas会增加**Batches**数量和带来**Canvas.RenderOverlays**函数的开销，反而会降低性能。

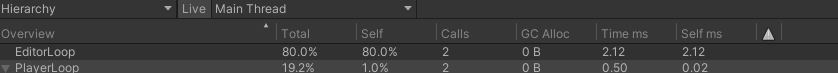
# Part I 不做动静分离

## D20S20

\_dynamic : Image数： 20

\_static : Image数： 20



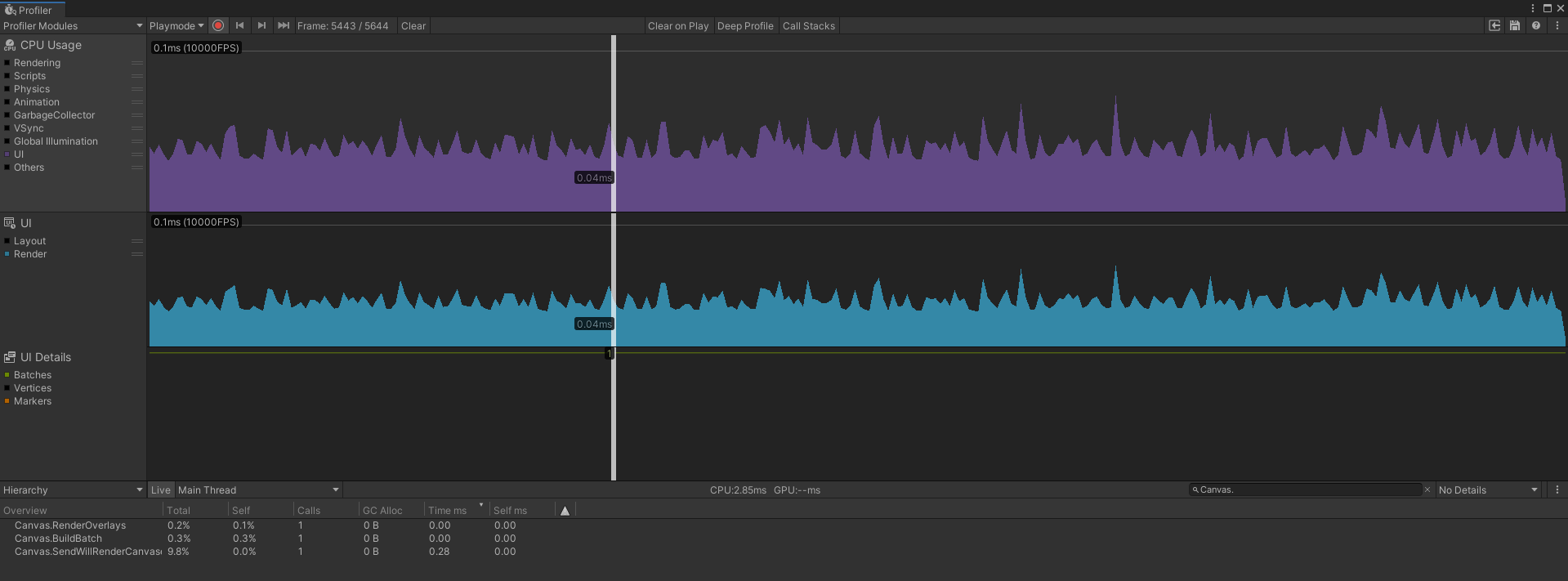


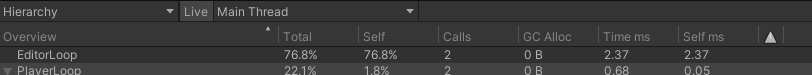
Batches : 1

## D20S100

\_dynamic : Image数： 20

\_static : Image数： 100



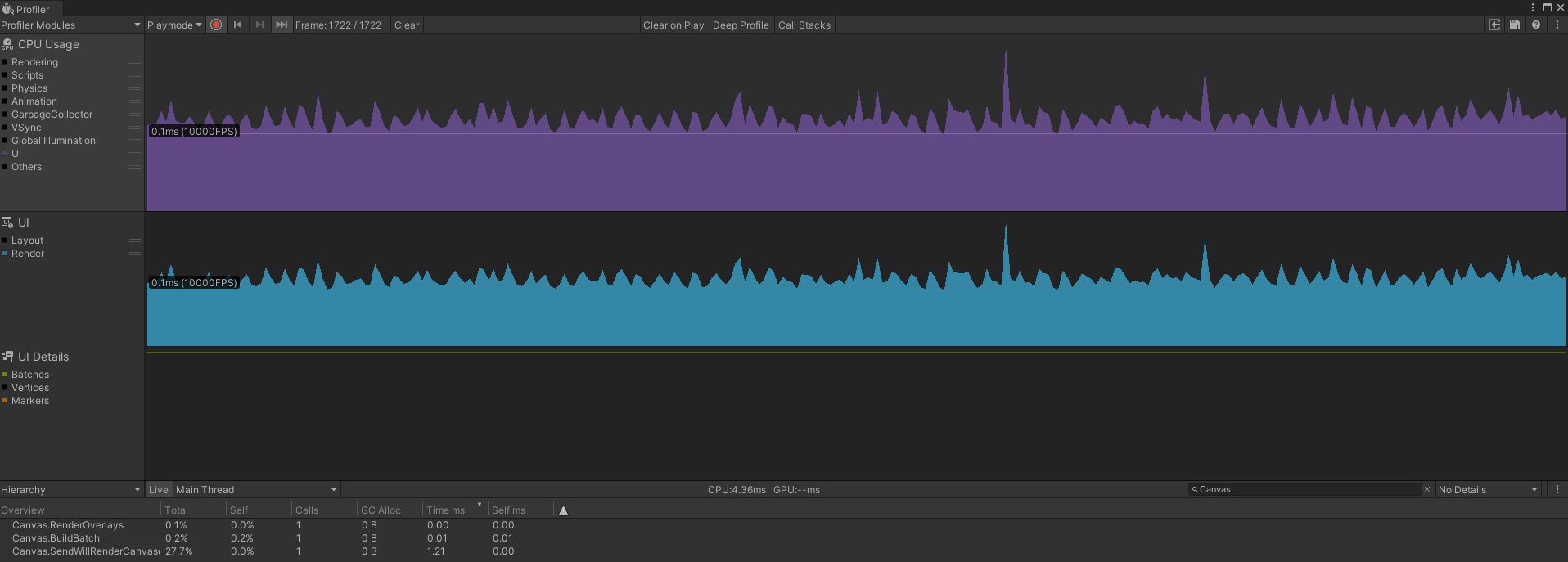


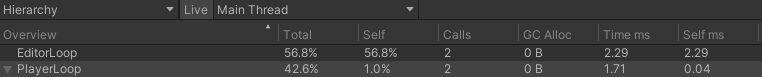
Batches : 1

## D100S20

\_dynamic : Image数： 100

\_static : Image数： 20



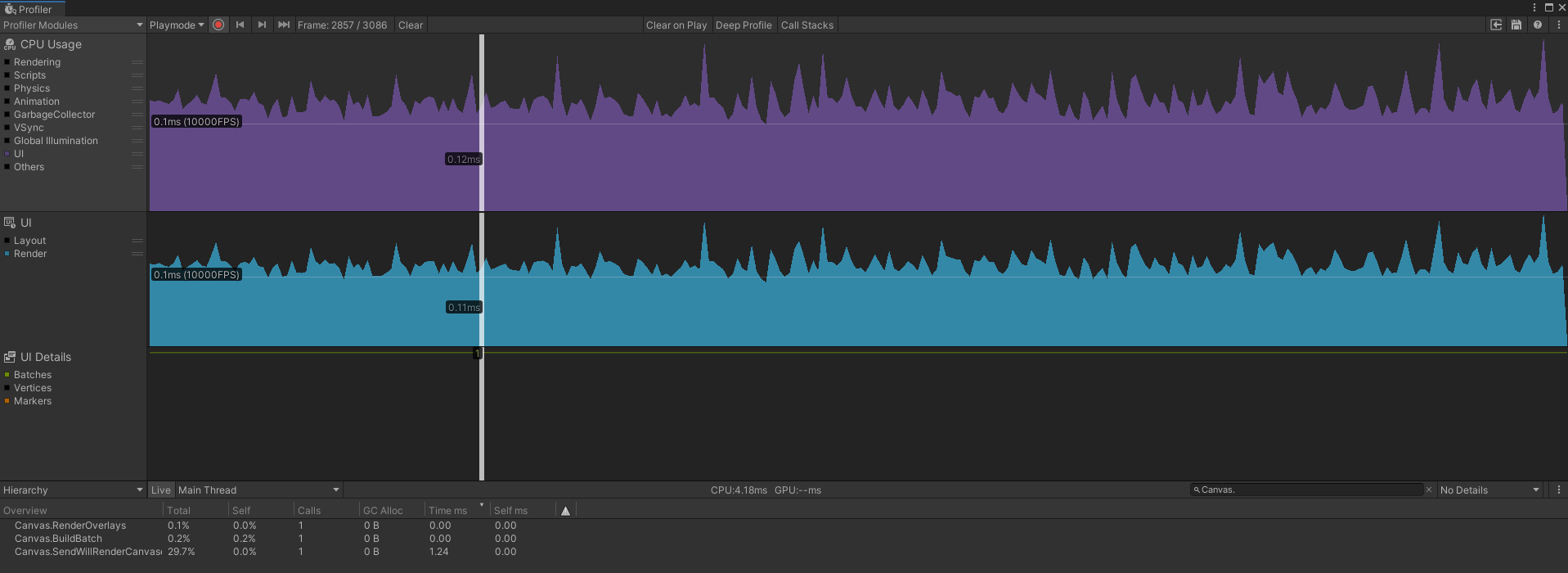


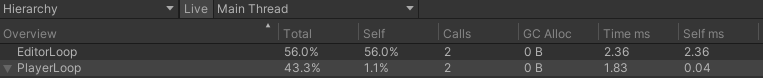
Batches : 1

## D100S100

\_dynamic : Image数： 100

\_static : Image数： 100





Batches : 1

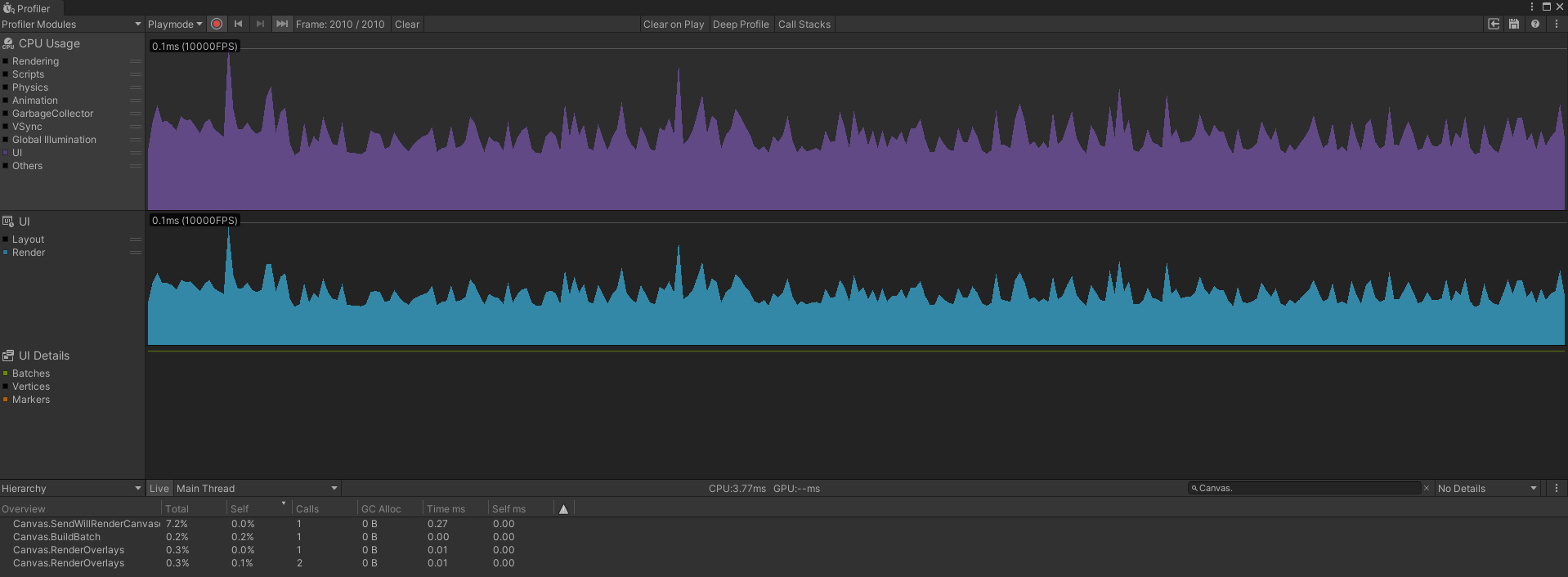
# Part II 做动静分离

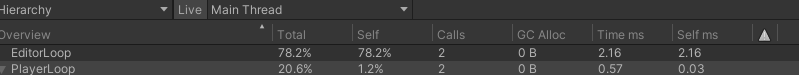
\_dynamic 挂载Canvas组件 \_static 挂载Canvas组件

## D20S20

\_dynamic : Image数： 20

\_static : Image数： 20



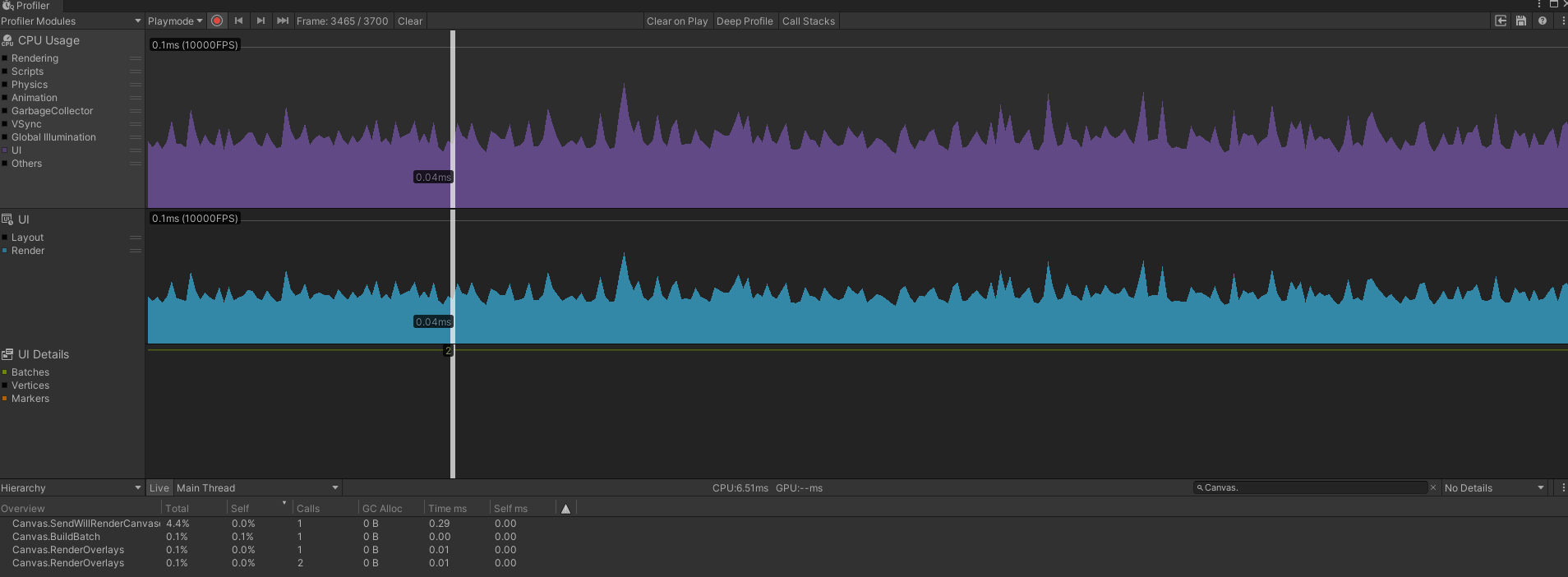


Batches : 2

## D20S100

\_dynamic : Image数： 20

\_static : Image数： 100



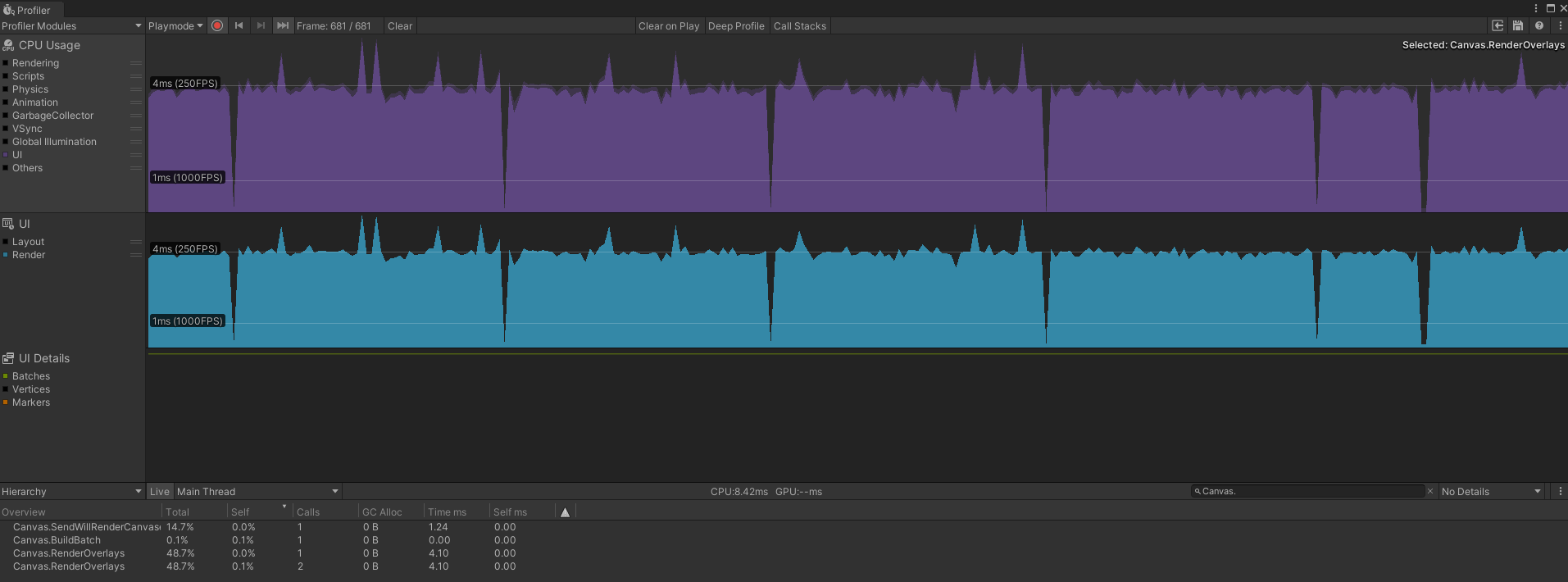


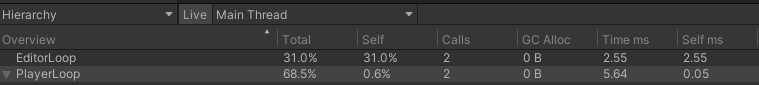
Batches : 2

## D100S20

\_dynamic : Image数： 100

\_static : Image数： 20



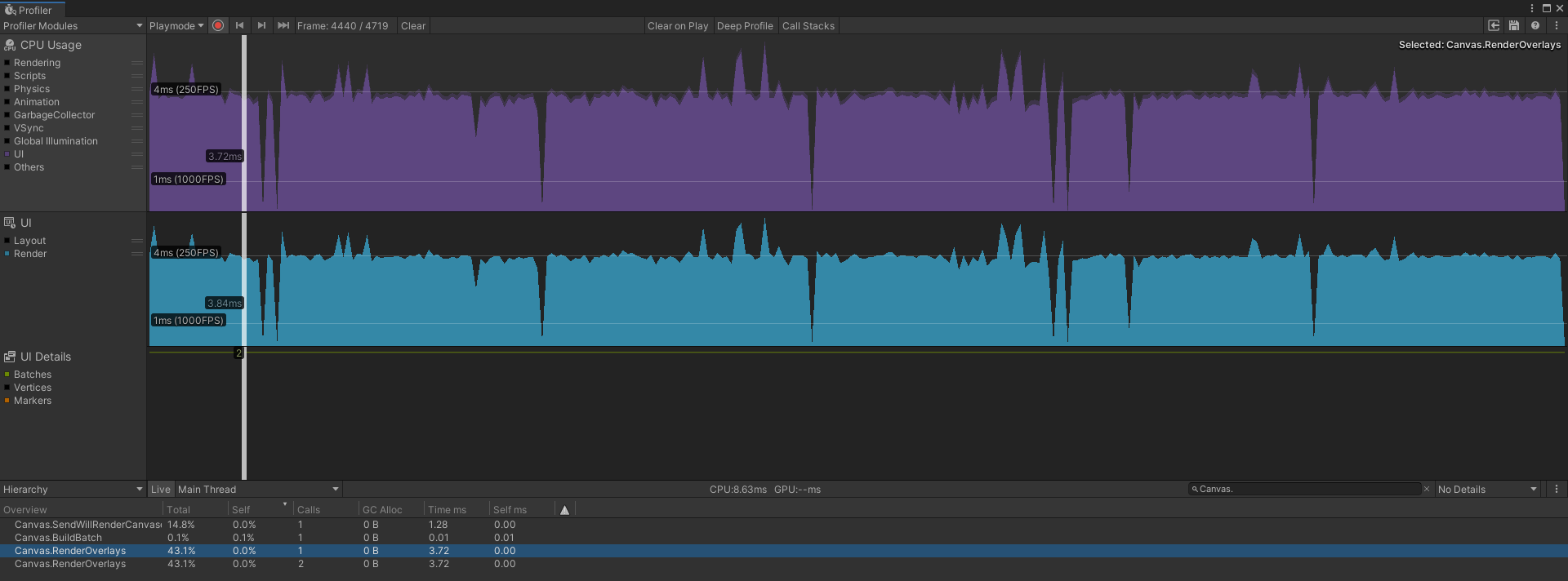


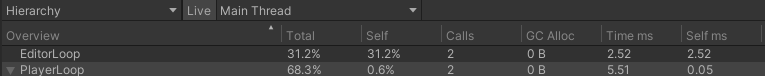
Batches : 2

## D100S100

\_dynamic : Image数： 100

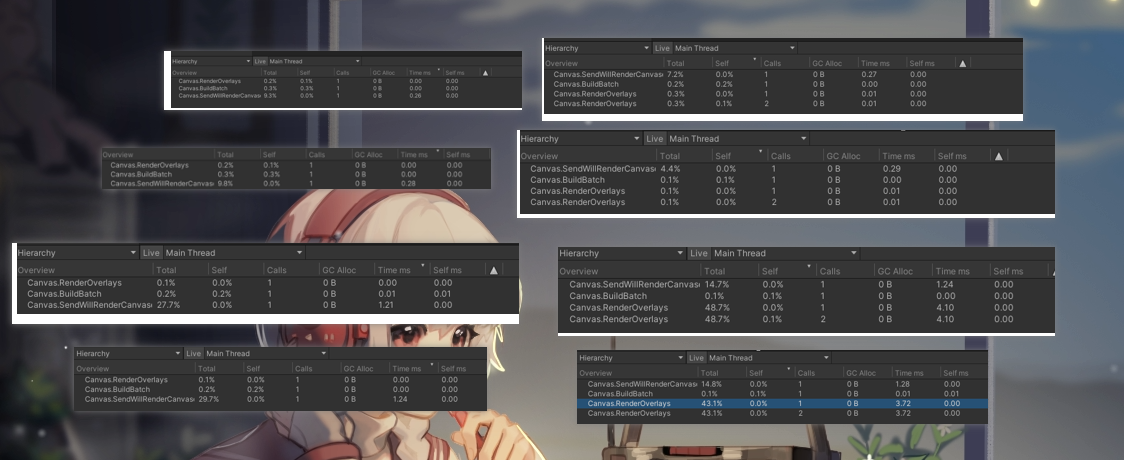
\_static : Image数： 100





Batches : 2

# 数据分析用途



# UIBattlePad分析数据

为了方便分析数据，临时给UIBattlePad添加Canvas组件。

由于UIBattleHp中有大量动态UI频繁更新，而他自身没有Canvas，故ReBatch会发生在\_\_dynamicRoot这个父节点上，从而影响对UIBattlePad的分析。通过嵌套Canvas来隔离，所以临时给UIBattleHp添加Canvas组件。

UIBattlePad未做动静分离的开销

当前结论：

Canvas.SendWillRenderCanvases()的开销在0.15ms左右

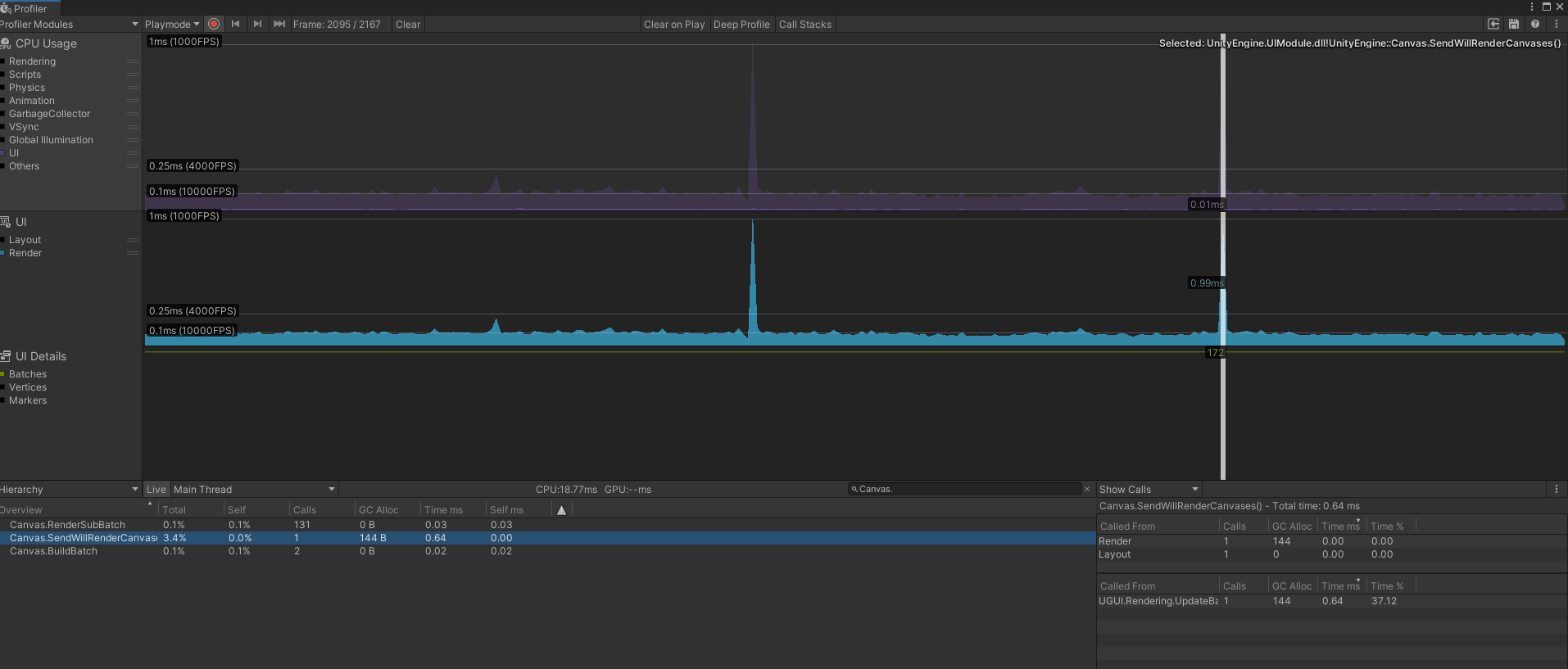
Canvas.BuildBatch的时间开销不超过0.03ms

给UIBattleHp和UIBattlePad添加Canvas来隔离UI元素变化引起的重新绘制，Canvas.SendWillRenderCanvases()和Canvas.BuildBatch 并未明显优化。合批的数量反而提高。

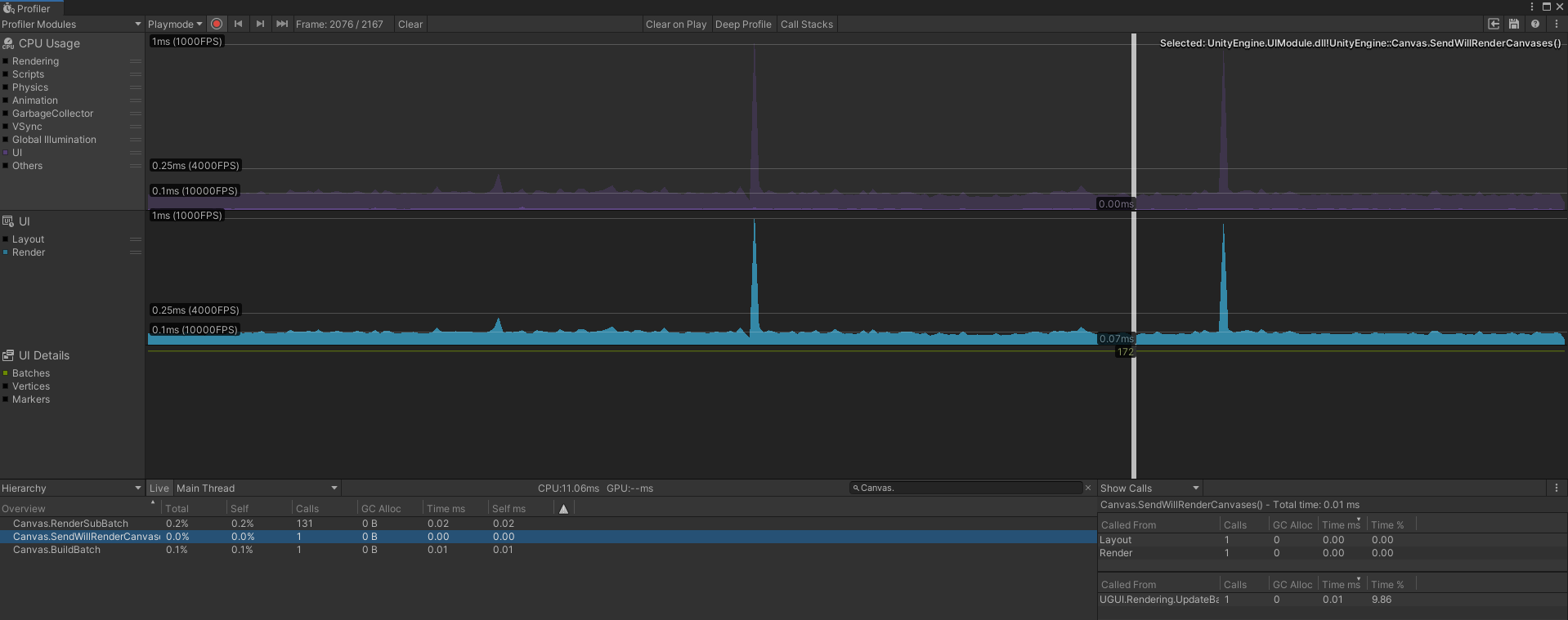
**采样：连续300帧**

**原本开销**

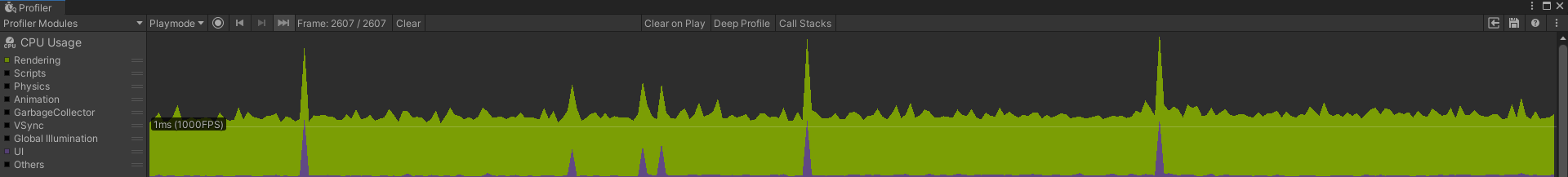
波峰：

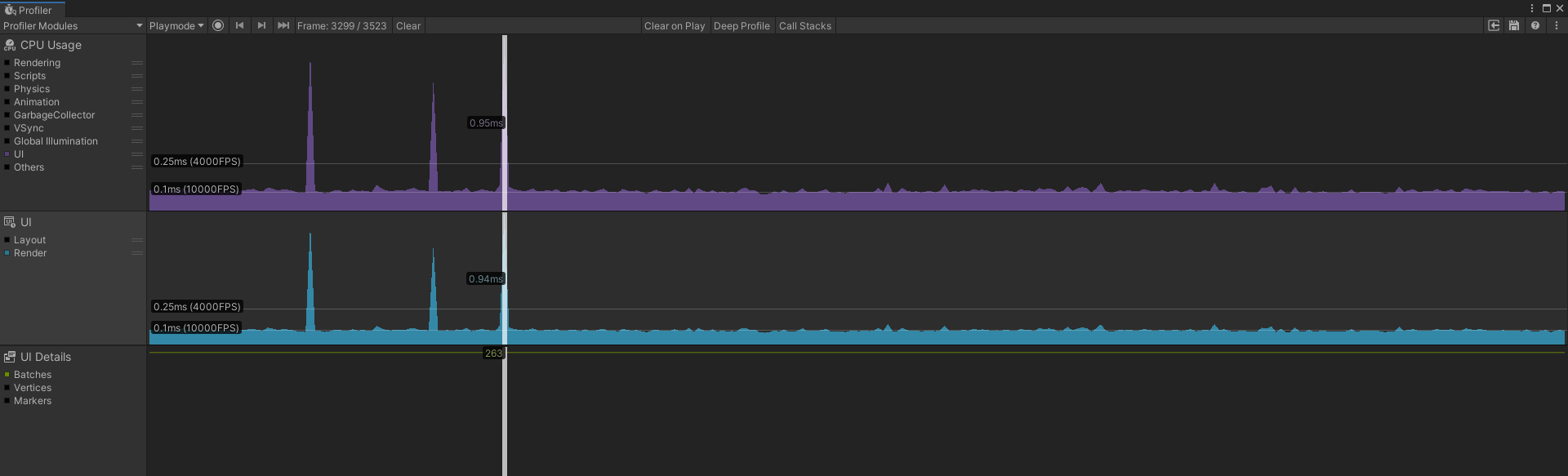


波谷：

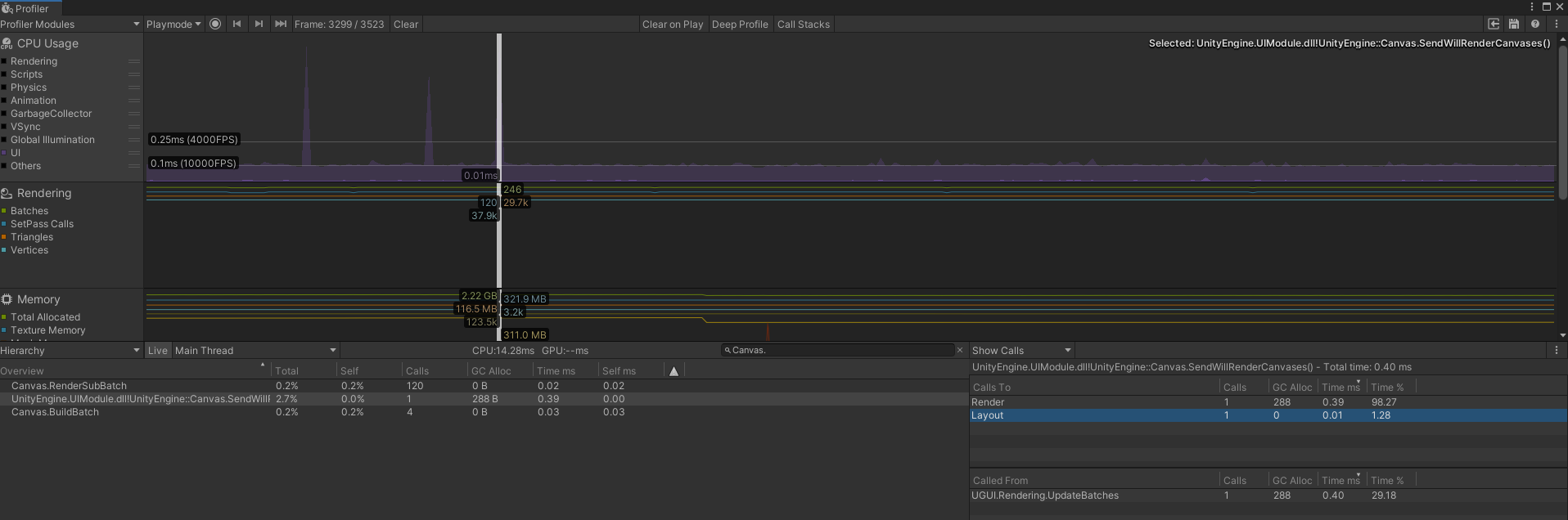


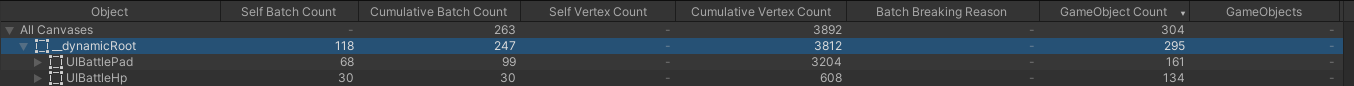
**UIBattleHp和UIBattlePad均添加Canvas后的开销**



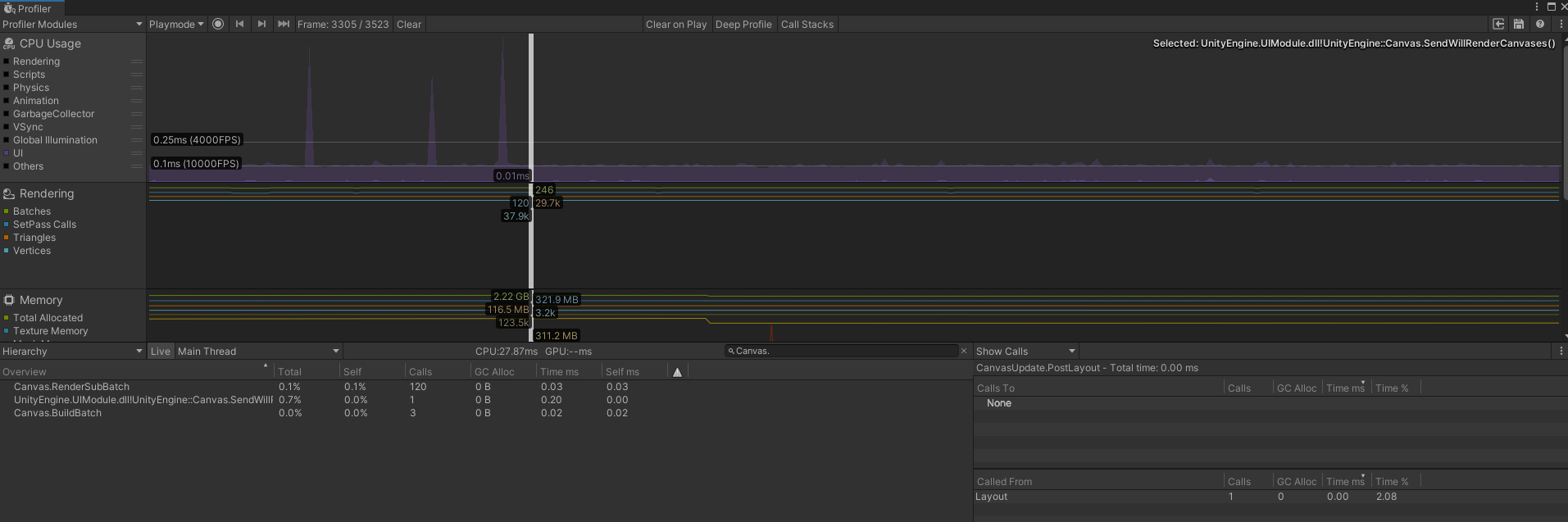


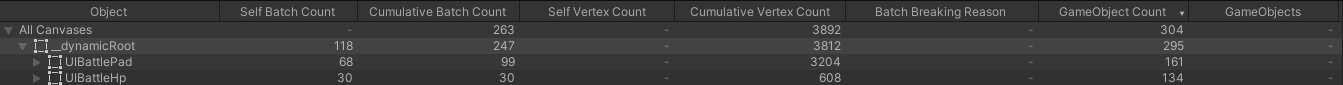
波峰

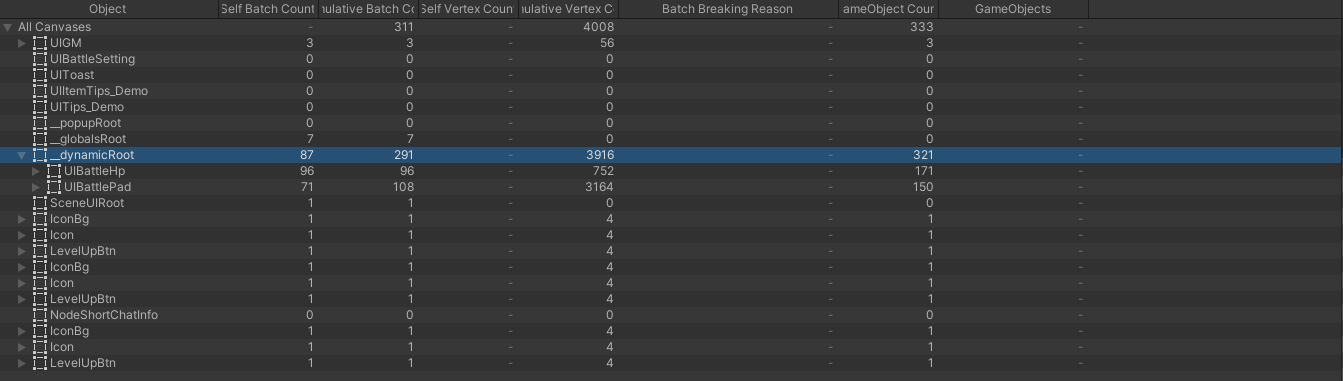




波谷







# 参考资料

https://www.xuanyusong.com/archives/4573

<https://learn.unity.com/tutorial/optimizing-unity-ui#5c7f8528edbc2a002053b5a3>

<http://www.manongjc.com/article/75137.html>

<https://docs.unity3d.com/Packages/com.unity.ugui@1.0/api/UnityEngine.UI.CanvasUpdateRegistry.html>

<https://blog.csdn.net/gaojinjingg/article/details/103565840>

<https://answers.unity.com/search.html?f=&type=question&redirect=search%2Fsearch&sort=relevance&q=Canvas.RenderOverlays>