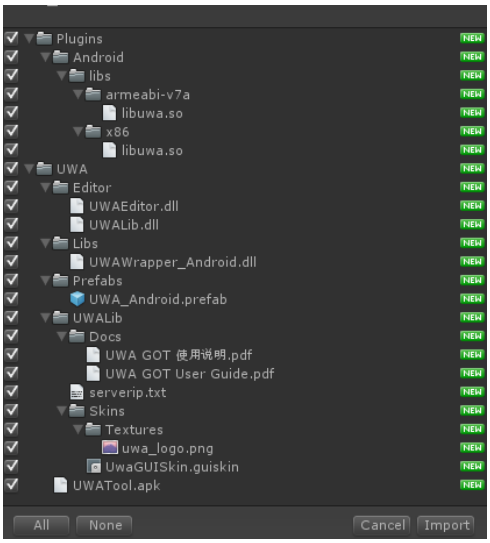
UWA GOT 使用说明

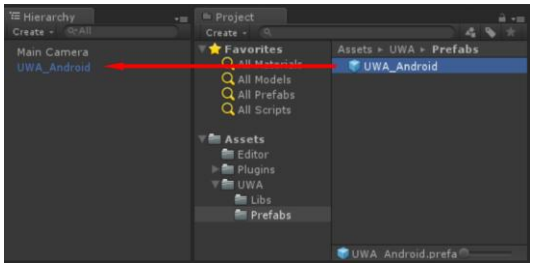
Game Optimization Toolkit 1.0.4 版本

GOT SDK 集成

1、 将 unitypackage 文件拖入到待测试的 Unity 项目中，当出现下图窗口时，点击“Import”按钮导入相关文件。

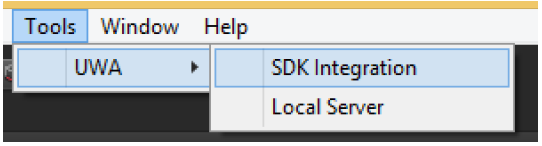


2、 在 Editor 中将 UWA/Prefabs 文件夹下的 UWA_Android.prefab 文件拖入到项目的首场景中，如下图所示。



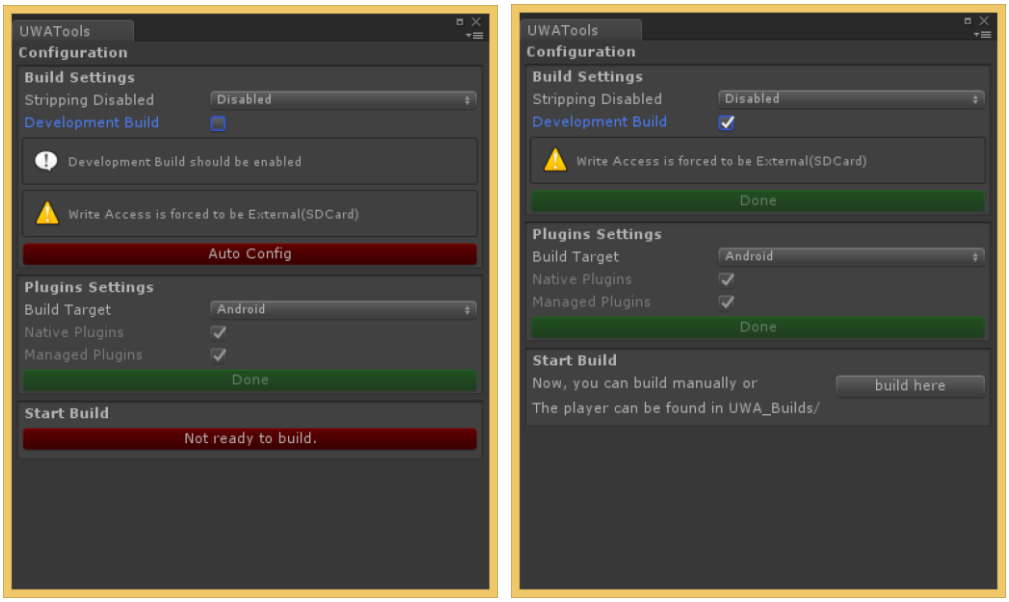
3、 如在 Game 视图的右上角出现如下图所示的 UI 界面，且无报错信息，则说明 UWA 插件已经集成到项目中。

4、 点击编辑器菜单栏中的“Tools->UWA->SDK Integration”选项，打开 UWA 工具栏。



5、 在出现的 UWA 工具设置窗口中，将 Stripping Disabled 设置为“Disabled”，并勾选 Development Build。用户也可以直接点击“Auto Config”按钮，进行一键设置，设置完成后，按钮会变成绿色，内容会变成“Done”。下方左图为设置前窗口视图，右图为设

置完成后窗口视图。同时，确保“Native Plugins”和“Managed Plugins”复选框为勾选状态，并确认 Write Access 已被设置为 External(SDCard)。



6、 发布游戏版本。建议用户直接点击 UWA 界面上的“build here”按钮来一键发布游戏，发布的 Apk 文件存放于 UWA_Builds/Android 文件夹中。同时，用户也可通过“Build Settings->Build”来手动发布游戏版本。

注意事项：

1、 目前该本地测试工具仅支持 Android 平台，设备无需 Root 权限。
2、 发布前确保 Stripping Level 为 Disable 状态。
3、 发布 Development 版本的 Apk 文件时，建议通过手动点击“build here”一键发布。通过 BuildPlayer 进行打包时，请添加 BuildOptions.Development 参数。
4、 请确认 Write Access 已被设置为 External(SDCard)。
5、 目前不支持采用 il2cpp 进行发布的项目。
6、 截屏的记录只支持 Android 5.0 或以上的系统。

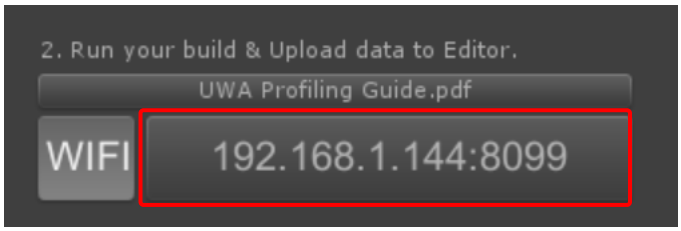
UWA Local Server

1、 在 UWA SDK 集成并发布后，您即可将其安装在 Android 测试机上并进行测试。

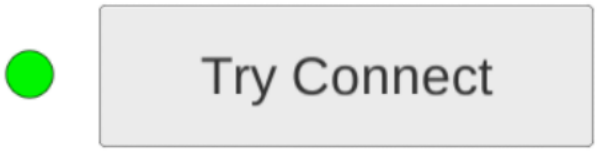
2、 打开 App 后，在界面右上角选择“Local Mode”，选择您想进行的测试类型。点击后，测试就会立即开始。当点击 Direct Mode 并使左侧按钮变绿后再点击某个模式，App 会自动退出，而在下一次开启时则立即开启测试。

3、 Stop 面板可以进行拖动，如果想结束本次测试，点击“Stop”即可。UWA SDK 会将搜集到的性能数据保存在设备本地。

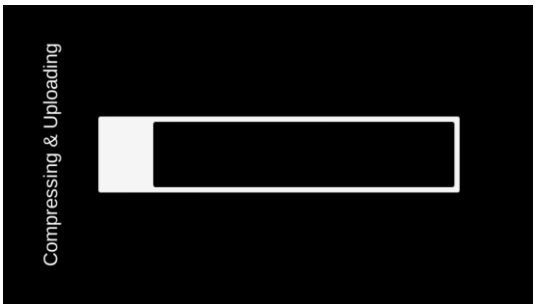
4、 点击编辑器菜单栏中的“UWA->Local Server”选项，点击面板上的“WIFI”按钮开启本地服务器。红框处即为您本地机器的当前 IP（端口固定为 8099）。若 IP 无法获取，则可以通过将 IP 写入 UWA/UWALib 下的 serverip.txt 文件中来手动配置。



5、 在测试手机上安装 UWA 文件夹下的 UWATool.apk 文件，该 App 主要负责将测试手机上的性能数据传输到 Editor 中的 Local Server 上进行分析。安装好后，打开 UWATool，输入 Local Server 的 IP 地址，点击“Try Connect”按钮，如果左侧圆点变绿，即表示可以连接到服务器。



6、 点击“Upload Data”，即可查看该移动设备上的所有测试数据，点击“Submit”，即可进入下图中的上传界面。对于无用的测试数据，点击“Delete”删除即可。当上传界面中的进度条结束，回到测试数据查看界面，即上传成功。



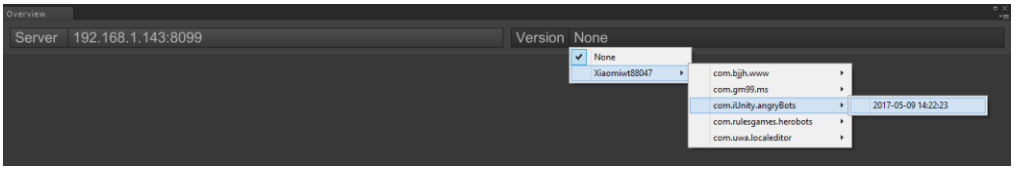
7、 在 Local Server 上，点击相关的按钮，选择您想查看的性能报告。

Performance Overview

点击“Overview”按钮，即可查看 App 运行时的整体性能情况。

1、 逻辑代码的 CPU 开销

（1） 在“Version”中选择您想查看的测试版本。



（2） 选择后，UWA 将为您载入相应数据并进行分析和展示。它主要包括以下视图：

i. **CPU 开销走势图**

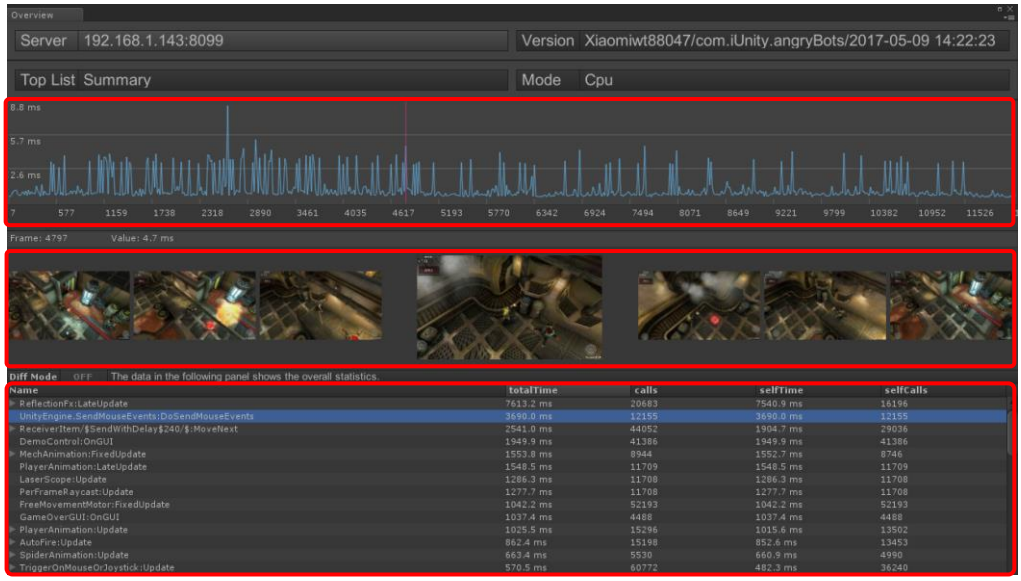
您可以选择任何一个函数，查看它在项目运行时的 CPU 开销。

ii. **截屏视图**

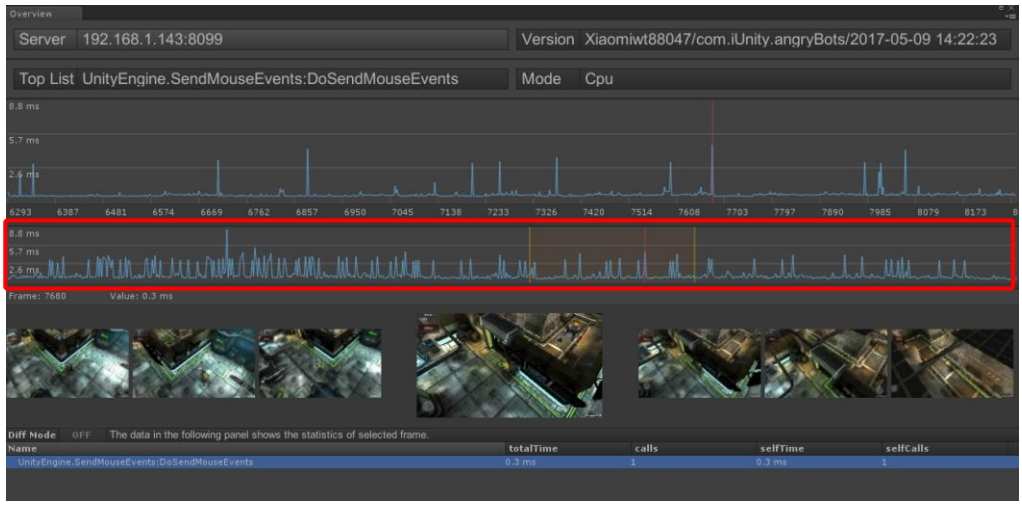
您可以在 CPU 开销走势图中选择任何一帧，截屏视图会随之切换到与其相对应的运行截屏。

iii. **CPU 耗时分析视图**

UWA 将逻辑代码的 CPU 耗时进行分析，并将最为耗时的代码展示在此。您可以通过 UWA API 统计指定的代码段的 CPU 耗时，具体用法见附录 1。



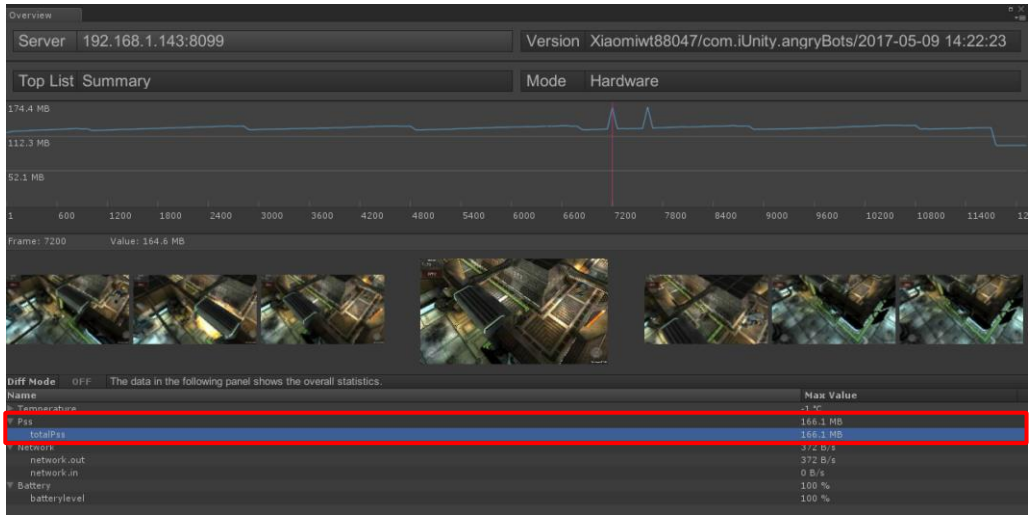
（3） 在此面板中，您既可以选择“Total”模式，查看逻辑代码的整体 CPU 耗时。也可以选择查看具体的逻辑代码。同时，您可以通过调整关注区域视图中的滑块，来重点查看您关注区域的 CPU 开销。



2、 硬件设备信息

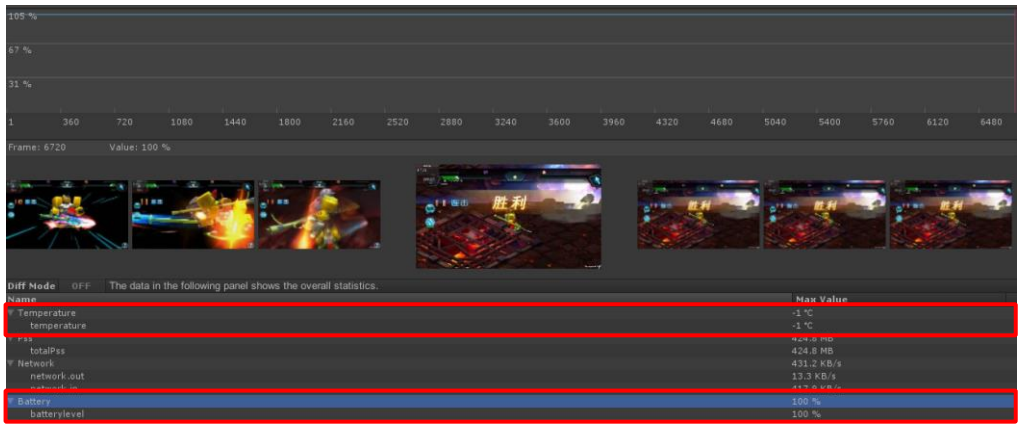
在“Mode”中选择“Hardware”，即可查看 App 运行时的硬件设备运行信息，主要包括：

（1） 硬件设备的内存信息。目前我们支持的设备为 Android 设备，所以主要显示 PSS 内存在项目运行时的走势。



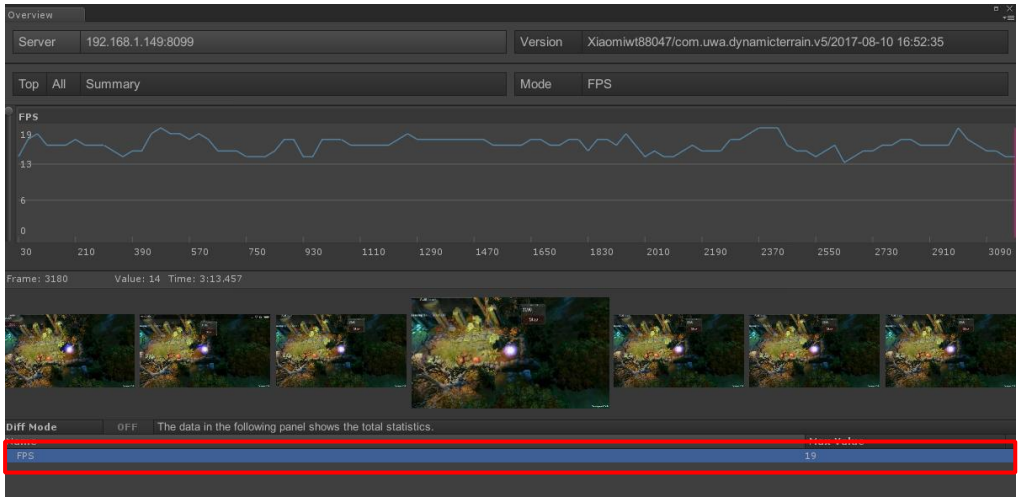
（2） 硬件设备的网络流量统计。Network.in 和 Network.out 分别表示硬件设备的网络下载和上传流量统计。

（3） 硬件设备的电量和温度。



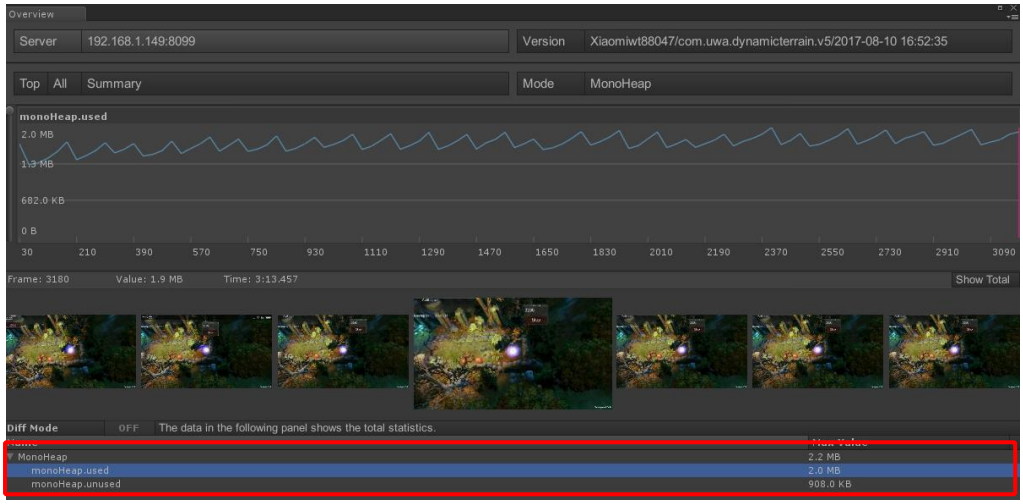
3、 FPS 信息

在“Mode”中选择“FPS”，即可查看 App 运行时的 FPS 统计。



4、 Mono Heap 信息

在“Mode”中选择“MonoHeap”，即可查看 App 运行时的 Mono 堆内存总量统计，包括了使用中的和未使用中的部分。

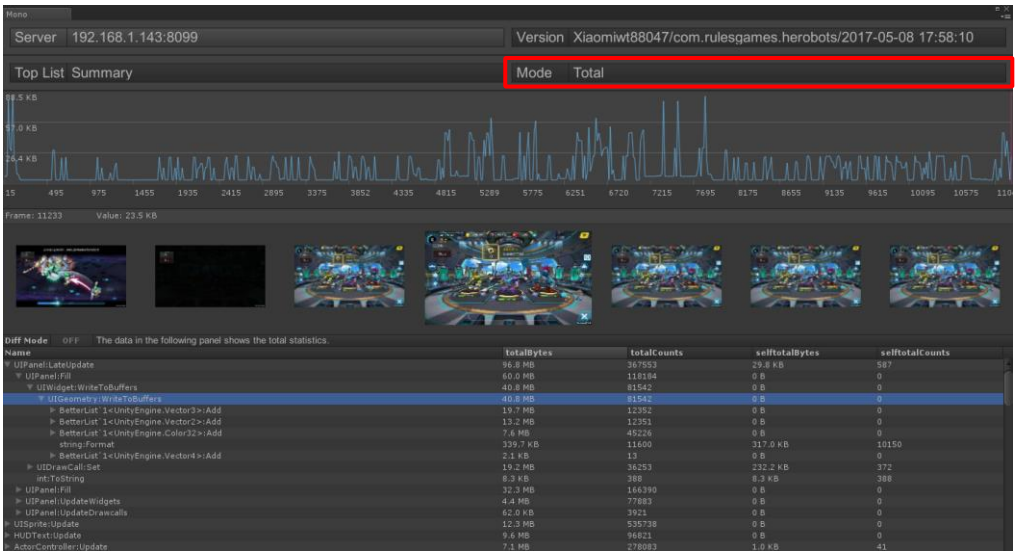


Mono Memory Analysis

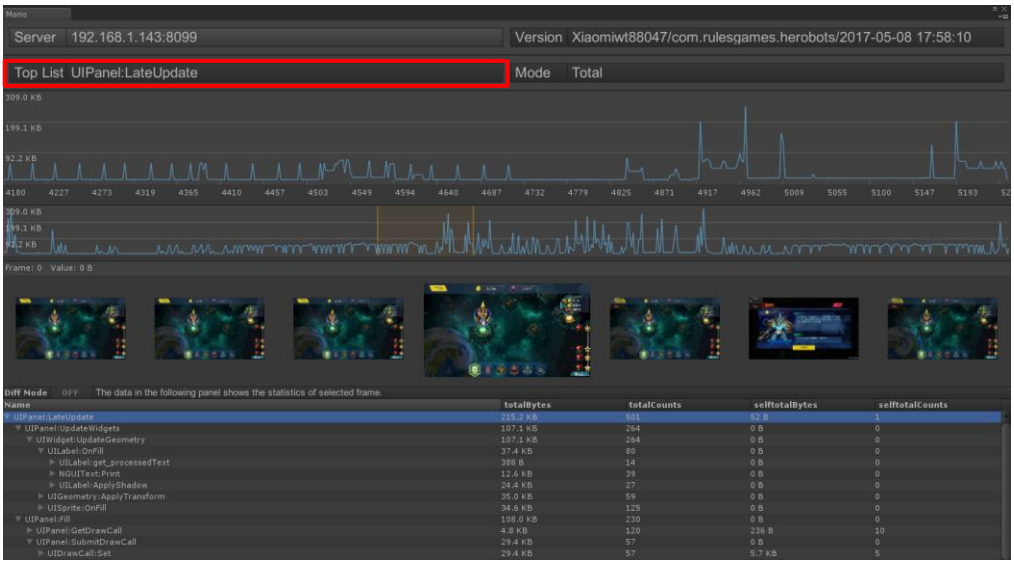
点击“Mono”按钮，即可查看 App 运行时的 Mono 堆内存分配情况。它主要包括以下功能：

1、 代码堆内存累积分配

（1） 在“Mode”中选择“Total”，您即可查看 App 运行时每个函数的总体堆内存分配情况；

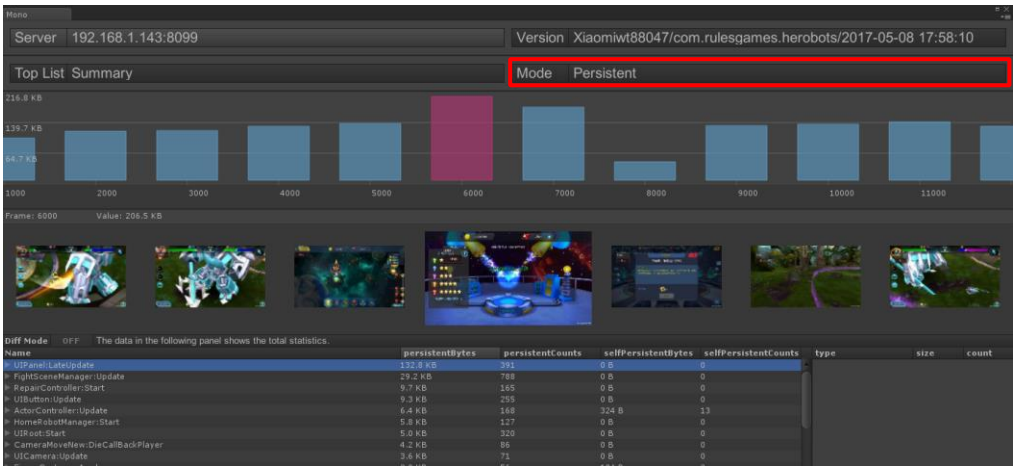


（2） 在“Top List”中选择具体的函数名称，您就可以看到相应函数的具体堆内存分配情况，并且通过与图表进行交互来查看任何一帧的具体堆内存分配。

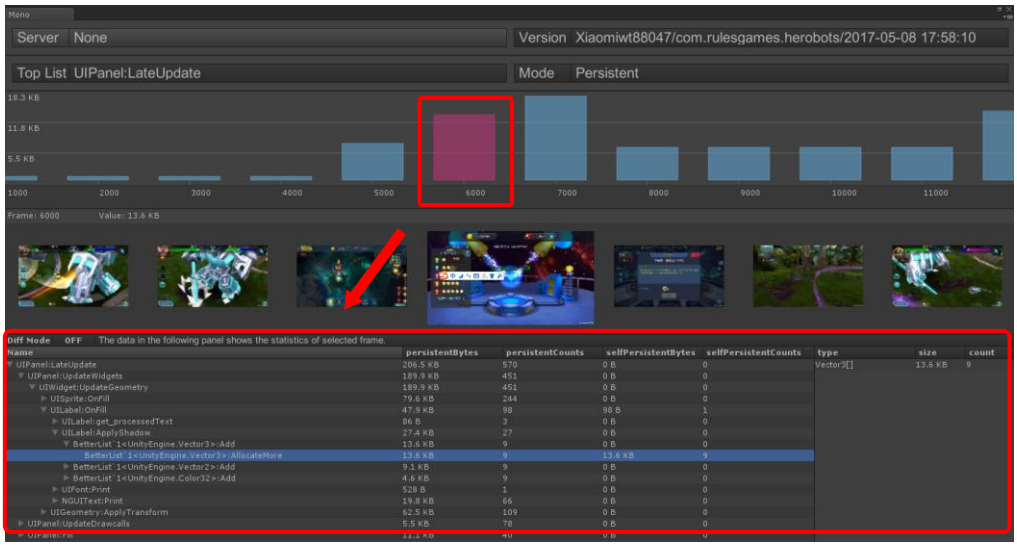


2、 代码堆内存泄露分析

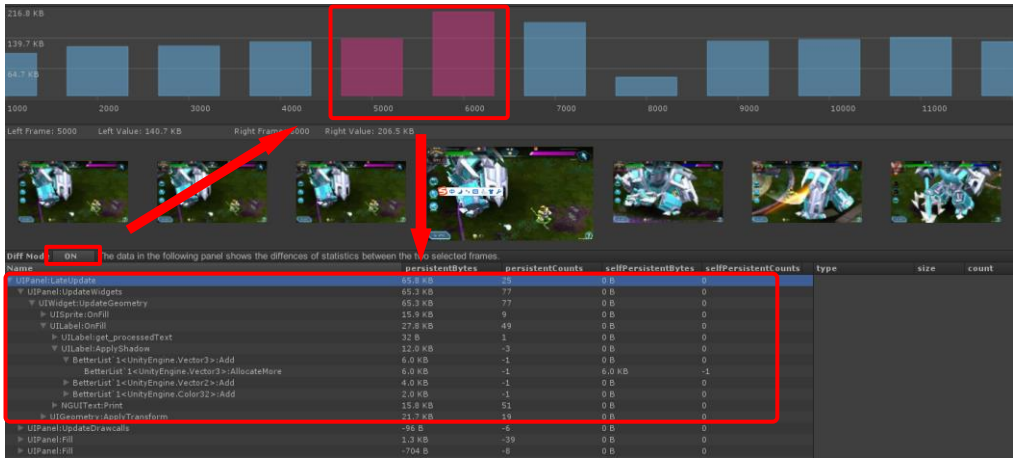
（1） 在“Mode”中选择“Persistent”，您即可查看 App 运行时每个函数在 Mono 中的真实驻留情况。UWA 默认是每 1000 帧分析一次 Mono 堆内存快照，将函数真实的堆内存驻留情况以柱状图的形式进行显示。



（2） 在“Top List”中选择具体的函数名称，您就可以看到相应函数的具体堆内存分配情况，并且通过与图表进行交互来查看详细堆内存驻留情况。同时，当 selfPersistentCounts 不为 0 时，点击可以查看由该函数生成的、驻留在堆内存中的变量类型。



（3） 在 Persistent 模式下，您可以比较两次堆内存统计的差异，从而来快速定位堆内存变化的出处。在“Diff Mode”中选择“ON”，即可开启该功能。选择任意两个柱状图，您则可以快随比较两次堆内存占用的差异。

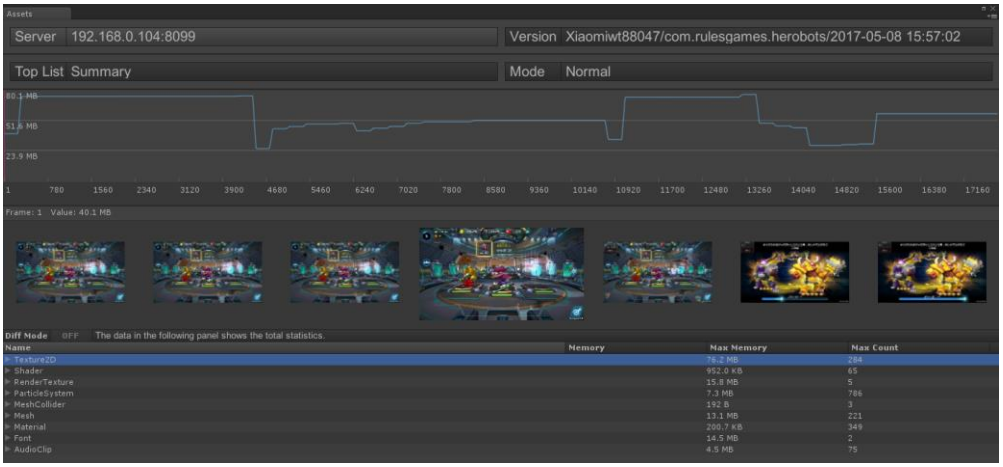


Runtime Asset Checker

点击“Asset”按钮，即可查看 App 运行时资源的具体使用情况。它主要包括以下功能：

1、 资源使用情况

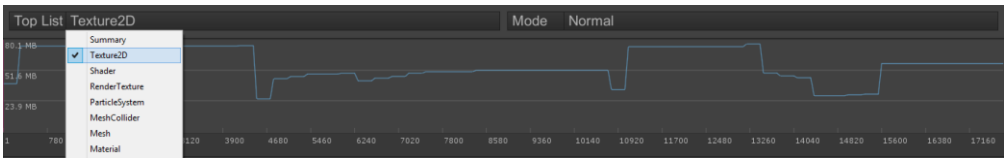
（1） 可以查看重点资源在项目运行时的内存占用情况。



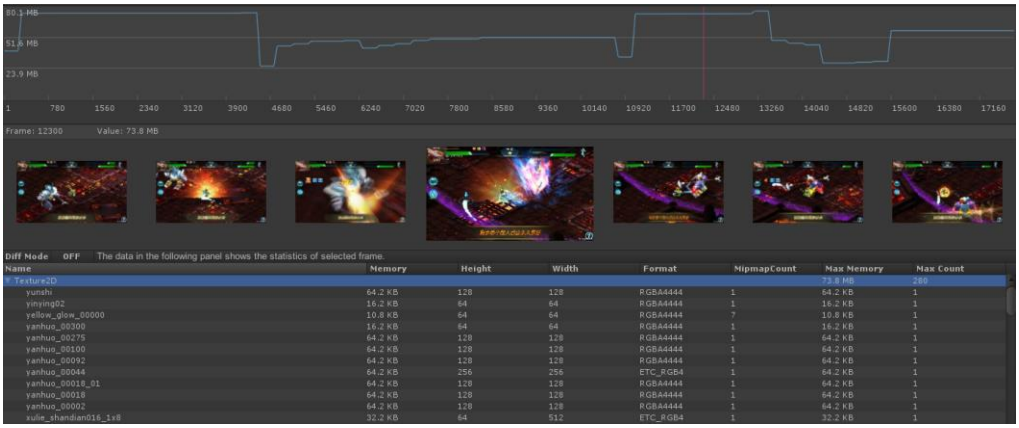
（2） 可以查看具体资源在项目运行时的使用情况。

2、 查看每帧中资源的具体使用情况

（1） 在 TopList 中选择您想查看的资源类型。



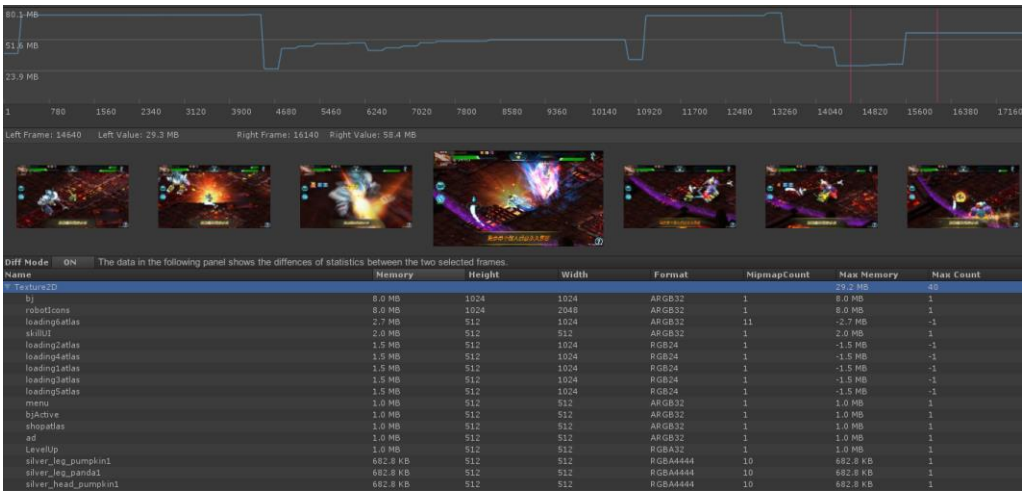
（2） 点击资源使用走势图，即可查看每帧该类资源或某个特定资源的具体使用情况。



3、 资源泄露分析

您可以通过比较任意两帧的资源变化情况，来分析是否存在资源泄露等问题。

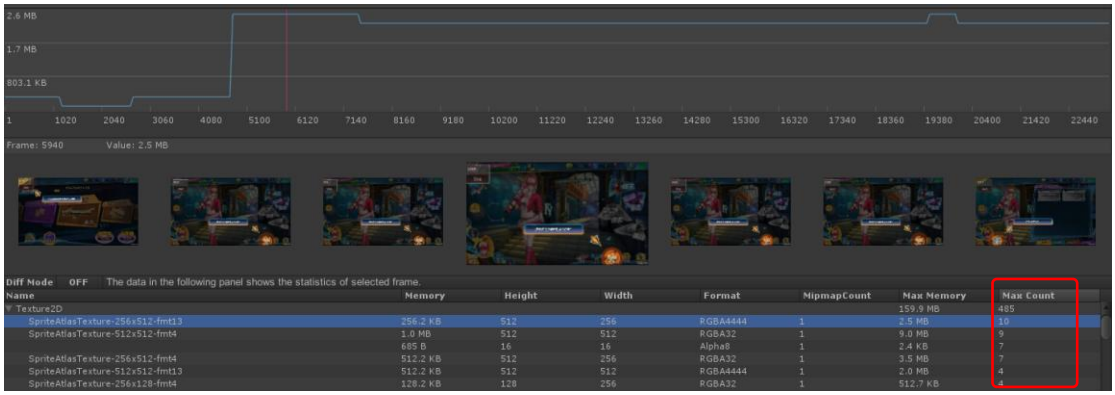
（1） 将“Diff Mode”设置为“ON”，选择资源使用视图中的任意两帧，即可查看资源的变化情况。



（2） 上图为第 16140 帧与第 14640 帧的 Texture 比较情况。其中，“Max Memory”中为正值的资源表示为第 16140 帧中的新增资源，而负值的资源则为第 16140 帧中的减少资源。通过这种比较，即可帮您快速定位具体的资源变化量和解决资源泄露等问题。

4、 资源冗余分析

项目运行过程中，内存中的资源很有可能出现冗余情况。对此，建议您详细查看资源数据展示界面中的“Max Count”数值， “Max Count”大于 1 的资源存在冗余问题的风险较高。Max Count 是指项目运行过程中，某一资源在某一帧中的最大资源使用数量。



注意：Max Count 资源数量大于 1，并不能 100%说明该资源存在冗余，也有可能是内存中确实存在两个资源名称、内存以及各个属性均相同的资源。因此，我们将 Max Count 大于 1 的资源称为“疑似”冗余资源。

附录 1：UWA API 的介绍和用法

UWA API 能够帮助开发人员统计自定义代码段 CPU 耗时，从而更快地定位脚本的性能瓶颈。

UWAEngine.PushSample/PopSample

public static void PushSample(string sampleName);

public static void PopSample();

参数 sampleName 表示自定义的函数标签，UWAEngine 会对 PushSample 和 PopSample 之间的代码段统计 CPU 开销，并在 UWA GOT 中的统计面板中进行显示，该 API 支持嵌套调用。其具体用法如下

```
UWAEngine.PushSample("MyCode");
// some code …
UWAEngine.PopSample();
```

最终在 Overview 界面中，可以看到自定义的函数标签，及其具体耗时（下图中 A~E 都是自定义函数标签）。

Name	percent	selfPercent	totalTime	calls	selfTime	selfCalls
Perf:Update	100.00 %	0.04 %	90.5 ms	3	0.0 ms	1
A	54.40 %	0.03 %	49.2 ms	11	0.0 ms	1
B	54.37 %	0.27 %	49.2 ms	110	0.3 ms	10
C	54.09 %	2.98 %	49.0 ms	1100	2.7 ms	100
D	51.11 %	30.92 %	46.3 ms	11000	28.0 ms	1000
E	20.19 %	20.19 %	18.3 ms	10000	18.3 ms	10000

请确保 PushSample 和 PopSample 是成对使用的。如果两者之间使用了 return 语句提前退出代码段（或者在协程中使用 yield return 提前跳出代码段），则会造成 PushSample 和 PopSample 的配对不准确，从而导致数据错误。

另外，请注意在同一帧中 PushSample 和 PopSample 的调用次数不宜过多。初步统计，在中低端的设备上，10000 次的调用会导致接近 50ms 的额外开销（这部分额外开销与 Unity 内置的 Profiler.BeginSample / EndSample 类似）。