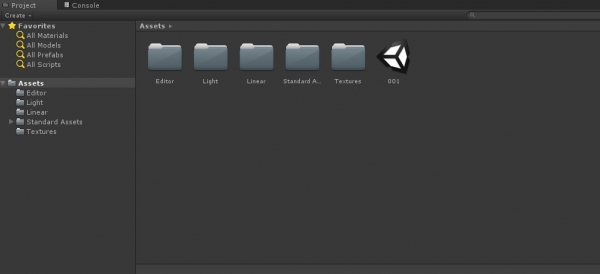
作者：Unity技术工程师柳振东

**上个月我们在Unity Connect上做了一期**[**移动游戏开发优化Q&A**](https://connect.unity.com/p/gan-huo-yi-dong-you-xi-you-hua-da-yi-zong-jie)**，Unity的官方技术工程师为大家解答了许多移动游戏开发优化的问题。今天，我们请到了Unity技术支持团队的柳振东，给我们总结了他平时工作中常见的性能问题。下面就进入正题：**

Unity技术支持团队经常会对有需求的客户公司项目进行游戏项目性能审查与优化，在我们碰到过的各种项目相关的问题中也有很多比较共同的方面，这里我们罗列了一些常见的问题并进行了归类，开发者朋友们可以参考下。

**资源导入**



**纹理没有压缩**

在很多情况下，美术会觉得纹理压缩后效果不理想。我们建议的是：可以把原图的分辨率长宽都扩大一倍，保持原有压缩格式。这样压缩过后的文件还是比不压缩的文件要小，并且视觉效果可以得到较大的改善。

**纹理导入设置中的 Read/Write Enabled 处于勾选状态**

开启纹理导入设置中 Read/Write Enabled，纹理在传到GPU之后，CPU端的数据也会一直保留在内存中。因为在移动端显存共享内存，会导致内存占用加倍。因此需要注意是否有需要在CPU端访问的纹理，比如：需要通过脚本获取纹理像素的情况下，就要开启纹理导入设置中的 Read/Write Enabled。

**模型文件导入设置中 Read/Write Enabled 处于勾选状态**

除了需要脚本中访问的网格，作为网格碰撞器中的网格，脚本中用StaticBatchingUtility.Combine静态合批的网格，以及粒子系统发射的网格之外，其它模型建议不要勾选此项 ，否则会在内存也保留一份网格实例占用内存。

**模型导入设置［Rig］选项页中Optimize GameObject没有勾选**

建议开启Optimize GameObject，这个选项可以把SceneManager中用于skinning的节点都去除，节省了场景节点树更新以及查询的CPU消耗，对于需要做挂点的节点可以添加到例外列表中。

**使用第三方音频插件时没有禁用Unity内置音频**

不需要使用Unity内置音频模块的时候，建议Editor中通过勾选Edit->Project Settings->Audio->Disable Unity Audio来完全禁用FMOD模块，避免不必要的CPU消耗。

CPU常见性能问题



**频繁调用的Camera.main**

建议脚本做好Main Camera的Cache。Camera.main实际为GameObject.FindGameObjectsWithTag(“MainCamera”)调用，主要因为引擎无法得知用户通过脚本设置的MainCamera，CPU消耗较高。

**脚本中大量UnityEngine.Object的判等操作**

建议改为用InstanceID来判断即Object. GetInstanceID，运行期间保证唯一。 因为Object的判等还有额外的耗时操作，而Int类型的判等就非常快速了。同理，使用Object作为key的数据结构也建议改用InstanceID做key。

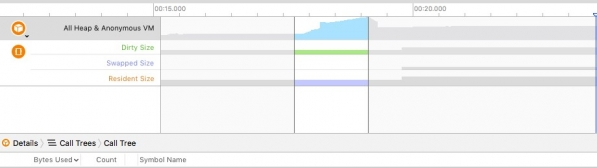
**用于查询操作的数据使用list数据结构**

List线性结构Contains的耗时非常高，建议改为hashset，hashtable之类的查询操作效率高的数据结构。

**加载资源时每帧从Assetbundle加载的Asset数量没有限制**

在场景内每帧从Assetbundle加载的Asset数建议限制在2到5个，数量高时耗时过长容易造成卡顿。

内存常见问题



**加载场景的时候有 一段内存尖峰**

常见的情况是有无索引的资源被加载进来，然后因为UnloadUnusedAssets被卸载掉。内存尖峰基本都是对游戏本身无用的内存，但是可能会因此造成游戏在内存紧张的机器上被强制关闭。

**静态索引导致的内存泄漏**

一些内存占用较大的资源如纹理,因为有静态索引而无法在切换场景或者调用UnloadUnusedAssets时被卸载掉，因此内存的泄漏量会随着用户切换场景的次数而增加。

**Mono内存池总大小与当前使用大小的差值较大**

这个值越大说明有越多的不必要内存池扩展，比如说在同一帧内有加载大量资源，实例化大量对象等，可能让内存池瞬间膨胀的操作。

**GC分配量较大**

项目Review过程中，除了CPU时间占用和内存分配量，我们还会留意脚本函数的GC分配。GC分配越频繁，量越大，由于Mono内存池可用内存不足导致的GC.Collect（造成卡顿原因之一）调用就越频繁，并且可能引起mono内存池不必要的扩展，因此脚本函数的GC分配量是既影响CPU也影响内存的重要参数。对于GC分配量。

我们**建议的参考数值为**：

* 对于基本每帧都会分配GC的函数，GC分配量大于2KB的建议都确认下是否有可能把临时变量抽取出来。
* 对于偶尔分配GC的函数，GC分配量大于10KB的建议都确认下分配的数据结构是否有优化空间。

GPU常见性能问题



**特效渲染的Pass数量较多**

一些特效的渲染可以合并到同一个Pass以节省GPU开销，另外RenderTexture在可以共用的情况下尽量共用。

**同屏面数过多**

同屏面数建议在20W以下，较优情况是控制在10W以内。

**UI元素在需要隐藏的时候使用了设置Alpha为0的方式**

实际上GPU依然需要对UI mesh进行渲染，建议不要通过设置Alpha为0的方式来隐藏UI。

**使用网格作为地形时，没有切分地形网格**

在网格顶点数很高情况下需要依靠硬件裁剪来剔除顶点，比较消耗GPU性能，建议按照大概的同屏可见范围来切分地形网格。

**UI元素过多依赖多层元素的混合来达到美术效果**

这样会造成较多的Overdraw，建议尽量通过预制纹理来做到想要的效果。

小结