# 优化思路

* 明确移动平台高中低档机器的各项数据的上限值，根据不同的机器，开启渲染效果和功能
* 优化主要是空间和时间上做平衡
* Unity在移动平台 一般CPU和渲染是瓶颈 所以多以空间换时间，来获得流畅的体验,有以下几点:

1. 针对CPU 一般就是降DrawCall 在游戏中DrawCall对CPU的影响远大于逻辑代码

250 unity能够支持的上限值

战斗场景 单人进场景 80(带UI) 留给特效 170 只支持5人同放大招特效

其中 主角使用高等级 别人使用低等级 再加上场景特效，高级装备

1. 针对渲染 减少模型顶点数量 贴图大小要控制
2. 针对内存 贴图是内存大户 以空间换时间重要的前提就是要控制内存 贴图就是大头
3. 制定好手机上适用的贴图尺寸，另外贴图要使用压缩格式 除了UI上有特殊要求
4. 针对游戏中的卡顿点，一般都是资源加载或者对象创建 采用缓存去解决
5. 还有就是GC也会造成游戏的卡顿，针对GC 就是C#代码要做好，隐式产生临时对象的用法要杜绝在Update中或者多次调用的情形下使用，否则极易触发GC也会致使MONO内存飙高.

有几个写法很容易出问题:

* 使用string += 做字符拼接会产生临时string对象很耗内存 使用stringBuilder代替
* String.format 的字符串拼接会有隐式的对象装箱和拆箱 避免在多次调用中使用
* foreach的使用 会产生大量临时对象 在代码中不要使用foreach用for代替
* 自定义的类如果使用object作为构造函数的参数，也会带来装箱和拆箱，要注意
* 缓存Unity的组件，不要频繁获取
* 能够复用的对象，多使用objPool
* 不要在Update或者循环中分配堆内存

1. 多人战斗情况下，功能选项 隐藏模型，隐藏特效, 隐藏头顶名称

* 特效优化

1. 特效DrawCall一定要控制 之前拟定了初版规则，可以根据测试表现作调整
2. 特效要制作高中低三档 技能特效主角使用渲染配置的档次，别人的特效一律使用低档
3. 特效中使用到的模型，要去检查，顶点过多要美术减面
4. 特效粒子的贴图单张不能超过128 合并成序列的贴图允许到512

* 场景

场景合并之后，内存消耗增加 可以视情况开启区域动态加载 可以降低内存占用但是会加骤游戏中的卡顿现象

# 注意事项

## 编译顺序

Utility

Engine

Common

DataCenter

Interface

EntitySystem

MapSystem

SkillSystem

Controller

ClientGlobal

GameMain

因为GameMain和bwtdev之间有循环依赖，所以GameMain编译完成后，让unityEditor加载编译bwtdev,然后再编译下GameMain

# 编辑器

目前来看，编辑器总体上不需要大改了，主要是小需求的完善