## 特效分层方案：

本次针对特效分层优化的目的：

对高中低配的机型显示不同优先级的特效，使各种机型跑游戏时能够运行流畅，维持较高，较稳定的FPS。

特效分层对于性能指标的提升：

1、减少DrawCall（CPU和GPU）

2、减少粒子发射器数量（CPU和GPU）

3、减少Monobehavior数量（CPU和内存）

4、减少Animation数量（CPU）

5、降低Shader复杂度（GPU）

6、减少贴图数量（GPU和内存）

7、降低填充率，overdraw，降低显存。（GPU）

区分高中低特效制作美术规范：

1、把复杂特效拆分成多个简单特效制作，便于特效分优先级管理

2、尽量避免使用大面积特效，减少填充率。

区分高中低特效制作流程：

1. 将复杂特效拆分成多个特效的组合特效。便于特效的分优先级管理
2. 在特效配置(EffectClip)中添加优先级属性，用于辨识特效自身的优先级
3. 根据系统配置信息，加载特效。在特效加载之前，读取该特效配置，并与系统配置比较，判断是否需要加载。（方案1判断逻辑：系统高配，高中低配的特效都加载。系统中配：只加载中配和低配特效。系统低配：只加载低配特效。方案2判断逻辑：系统高配，加载高配effectlist，系统中配，加载中配effectlist，系统低配，加载低配effectlist）
4. 如果有无法拆分的特效，可以在特效制作时添加标记，标记特定GameObject从属的层级，保存配置时把相对路径保存到配置中，在加载特效后，按照配置把不该显示的Gameobject隐藏，特效打包时删除该脚本。（优点，不会添加mono脚本，没有额外消耗。缺点，配置数据会变大因为存入了路径字符串，操作先对繁琐，需要美术手动添加脚本，而不是在编辑器的GUI中编辑的）
5. 编辑器支持：程序提供GUI编辑特效的高中低配置逻辑。
6. 程序提供单个特效下的不同Gameobject区分高中低配的脚本，需要美术去挂载。

分组后的特效播放流程图：

方案1:



方案2:



后续优化：

可以考虑粒子系统数据化，即保存粒子特效的数据，在游戏中做一个粒子系统的缓存池，如果有新的粒子特效需要加载，则取到缓存池中的粒子系统，把该特效的粒子数据赋值给它。

待讨论的问题：

1. 程序可以提供在特效加载后更换材质及改变shader参数的支持。如果美术需要可以提出相应需求。
2. 粒子发射器的粒子数量限制配置。（高中低配的粒子发射器发射粒子数量不同）
3. 分目录管理特效资源，每个特效根据不同的配置需求制作三个不同的资源（公用材质和贴图），放在高中低三个目录下。如：a特效有三个，分别在高中低档时表现，低档a特效放在低档目录，高档a放在高档目录。优点是：资源便于管理，同一个a特效可以在不同档次的技能组表现。缺点是：工作量大，资源数目增多。
4. 场景特效，Cutscene特效是否有角色特效制作方式一致。

总结：最终确定使用方案2制作流程。