1. app 包体优化。比如，做了插件筛查无用资源、在可接受范围内压缩 png 、压缩声音音效降低比特率使用单声道等、固定字使用图片字、如果一定要使用 ttf ，可以对 ttf 进行编辑删除没有用到字或者生僻字，序列帧改骨骼等。 2. drawcall 优化，要使用图集，调整渲染顺序尽可能地让引擎库批次渲染。 3. 内存管理，资源尽可能地分模块管理，动态施放暂时不使用的资源。 4. 节点池，对于常用的节点使用 nodepool 5. 服务器慢查询优化 6. 后台打开速度查询速度优化 7. cpu 优化，对于需要大量计算的操作，能预制的可以先预制出来，用内存换时间，对于复杂需要实时运算的业务，可以独立线程运算，或者独立进程甚至独立服务器出来运算。 8. 内存优化，基本上出现内存过高都是有内存泄漏存在，检查 gc ，和是否是未 delete 的行为，框架开发者要考虑尽可能地让业务开发者不使用 new 来处理业务 9. 业务上来以后的集群支持
2. app 包体优化。比如，做了插件筛查无用资源、在可接受范围内压缩 png 、压缩声音音效降低比特率使用单声道等、固定字使用图片字、如果一定要使用 ttf ，可以对 ttf 进行编辑删除没有用到字或者生僻字，序列帧改骨骼等。 2. drawcall 优化，要使用图集，调整渲染顺序尽可能地让引擎库批次渲染。 3. 内存管理，资源尽可能地分模块管理，动态施放暂时不使用的资源。 4. 节点池，对于常用的节点使用 nodepool 5. 服务器慢查询优化 6. 后台打开速度查询速度优化 7. cpu 优化，对于需要大量计算的操作，能预制的可以先预制出来，用内存换时间，对于复杂需要实时运算的业务，可以独立线程运算，或者独立进程甚至独立服务器出来运算。 8. 内存优化，基本上出现内存过高都是有内存泄漏存在，检查 gc ，和是否是未 delete 的行为，框架开发者要考虑尽可能地让业务开发者不使用 new 来处理业务

在飞升开发过程中，遇到的困难比较多。当时cocos2dx的版本很低，很多功能都不完善。比如没有异步加载，引擎自带的异步加载是个假异步。不支持freetype，一行文件就会生成一个bitmap，性能差，占内存大。经过查阅资料，使用share context做到了真正的异步加载，给引擎集成了freetype。 当时最好的手机内存只有512m。对于资源管理和包容量要求也很高。也做了很多优化处理。

● 解决线上与开发中遇到的各种问题 ● 面试与筛选技术人员 ● 对新员工进行技术培训 ● 定期技术与总结分享 ● 解决开发中团队中遇到的技术难题与瓶颈 ● 进度管理，项目风险评估，技术方向把控 ● 自己负责的游戏模块业务开发 ● 美术风格讨论商定 ● 核心难点业务攻坚 ● 专业打杂，哪里需要去哪里