# 其他杂项交接

## UI相关

1. 我们用的FairyGUI，不是基于最新的4.3.0开发的，所以有一些编辑器中自带的功能代码没有；
2. 我们扩展了FairyGUI的功能，主要是一个不完善的RTL功能（虚拟列表，单行，溢出会出问题，这种时候建议自动翻转数据源），滑动条加了一个FreezeValue，用来做上限等，同时修改了底层阿语的一些适配问题；
3. 通过开启了FairyGUI的Batch，可以用动态批次合并，但是会增加CPU消耗，同时可能导致动效层级不对；
4. 所有的FairyGUI中的图片，如果在icon开头的包里，都可以用Editor的工具生成UIAtlasInfo表，这样就不用加路径，可以直接访问了；
5. FairyGUI中，经常会由于Destroy导致一些报错，遇到这些报错记得检查销毁接口；
6. 基于上面原因，在UI中用Timing之类的脚本，很容易产生空指针错误；
7. FairyGUI的动效，有很多问题，需要重置状态之类的，所以最好让策划拼的时候能够在第一帧把所有的状态都重置一下，避免出现问题；
8. UI我们都是同步加载，打开单个很大的UI界面是会很卡，为什么选择同步加载，是策划要求，从红警1时代就是这样做的，所以策划说卡没事，其实异步加载效果好很多，但是要处理异步时穿透的各种问题；
9. 同一个组件，如果被不同的扩展类注册，有可能导致出错，举个粒子：当A组件被注册为ClassA时，在一个列表中，创建了2个，那么他们都是ClassA，这时候打开界面B，又被注册成了ClassB，这个时候回到界面A，换了一个tab，list需要3个元素，就又创建组件A，这个时候他注册的是扩展类ClassB，但是代码当作ClassA来用，就报错了；
10. 接上一个，最好的UI管理就是让策划自己把通用组件做好管理，不乱用；
11. FairyGUI可以自动生成代码，参考我之前写的教学Demo，能够一定程度的提升代码开发效率，但是也需要注意，组件的命名等问题；
12. FairyGUI跨包调用的组件，自动生成代码没有办法实现扩展类，都默认生成为GComponent，这个如果要优化，自己写导出代码，不要用FairyGUI自带的，这个github上有例子；
13. 我们现在使用的FairyGUI的UIPanel模式，主要是为了兼容Unity之前的Prefab开发思路，其实完全可以不用，直接用组件扩展或者Windows，都非常好，以后可以优化；
14. UI中的列表，尽量都是用虚拟列表，提高效率；
15. 能不销毁就不要销毁界面，这样可以减少很多的GC和CPU时间；
16. 阿语中，文字有可能因为加入了一些特殊符号，比如冒号等，导致显示错误，最好去掉；
17. "UIBranch": "阿拉伯语"，ArchivedData文件可以通过加入这一行，实现文字是中文，但是UI结构是阿拉伯语，方便测试阿语UI；
18. 如果发现场景中，每次打开一个UI界面，就多一些未激活的节点，很有可能是有内存泄漏，比如打开界面就创建一个组件，关闭的时候没有释放，或者创建的时候没有判断之前有没有导致问题；
19. 游戏中有一个远程贴图加载功能，如果给GLoader传的值是http或者https开头，就会去网络上下载图片，放在本地，这个可以用来线上的时候热更新一些贴图；

## 加密相关

1. 现在只有打包机可以打包，其他的电脑可能因为加密问题，导致打出的APK包被加密了，找杨桐给你换一个加密版本就可以解决这个问题；
2. 加密有时候会掉线，虽然本地看没问题，但是其实服务器那边掉了，这个时候SVN就没法访问，重启一般就好了，不行找网管；
3. 不能使用Git管理项目，SVN服务器也必须搭建在公司服务器上，不能私自搭建；
4. 不要把自己的代码上传到github等上面，不要在公司用各种同步关键；
5. 不要用自己的翻墙软件，用公司的，安全一点；
6. 加密有时候会抽风，导致加密了PackageManager里的代码，这个时候，你先恢复加密，然后移除本地的PackageManger里被加密的包，然后删除本地缓存（不然他会直接拷贝过来，还是错的），重新下载就对了；

## 项目优化

1. 如果说启动速度慢，是因为刚开始的场景加载的图集关联的太多了，可以让第一个界面只引用少量的，比如1-2张图集，可以加速；
2. 优化代码的可以考虑用对象池，优化对象，以及可回收的集合类去优化；
3. AB后期如果要做资源优化和内存管理，可以参考刘健之前让他做的Demo；
4. UI优化才是游戏优化的大头，但是非常难搞，可以先从删除不用的组件，贴图开始，减少图集和包的数量，这样减少浪费；
5. 现在的图片都用了ASTC8x8格式，这个格式比ETC2少四分之三的内存和包体占用，非常好，缺点是画质差一点，ASTC6x6画质和ETC2差不多，但是由于这个巨大的提升，之前郑文也看了觉得没什么区别，就用这个了；之前也上线统计过数据，98%-99%的玩家都支持这个，如果运营要数据，也可以找服务器看，具体的key在LoginCommand中，搜索astc就能找到；
6. 一般情况下，崩溃就两个原因，报错和内存不足；
7. 报错一般看log，一些空指针，数组越界都会报错，在32位下会崩溃，64位不会崩溃，但是报错；
8. 内存不足这个只有猜了，但是减少一些对象的创建，回收对象都是好的解决方法，不行就加入释放内存的逻辑；
9. 上面说到的刘健的Demo，可以完美解决FairyGUI的引用问题，但是对我们的代码结构有影响，可以尝试接入；
10. 我们现在的UI优化，是当有一个全屏界面，或者带模糊底的非全屏界面打开的时候，就把他层级之下的UI都移走移的很远，这样Unity就不渲染了，同时也把内城和大地图的相机的CullingMask都关闭，这样减少了很多的DrawCall；
11. 游戏逻辑体的DrawCall优化，基本上就用的是Unity的动态批次合并；
12. 以后如果做大规模的战斗，可以考虑用之前的GPU Instance的例子去优化；
13. 粒子也是很容易打断批次的东西，同时粒子需要借助GameObject才能实现逻辑，也很耗性能，可以参考网上一个脱离GameObject的粒子系统，可以优化；
14. 代码架构的优化才是真正的优化核心，只有代码架构清晰了，整个工程干净了，你才容易找出来，哪里有问题；
15. 每天的自动打包，会帮助解决BUG，比如一个BUG，昨天没有，今天有，那么大概率就是昨天提交的代码或者资源导致的；
16. 之前用过各种自动测试工具，但是由于我们的游戏逻辑变化太大，写自动测试脚本非常不方便，一天写10个，然后第二天就全废了，所以没搞，后期可以偷偷搞搞试试看；
17. 邮件模块如果要优化，可以考虑把邮件存在本地，参考万龙，只有收藏的邮件或者重要的某些类型邮件存在服务器，其他的收到之后就直接发送删除协议给服务器，然后缓存到本地，可以减少很多流量；
18. 目前我们的代码，AB包都没有加密，老板没提我们不做，如果有一天要做了，可以考虑以下方案：AB加密用offset或者异或加密，防君子不防小人，这样不影响性能；代码加密可以用一些第三方的混淆工具；
19. 工程中，还有一些表格是旧版本json配置的，自己写的加载，没有用雷神的工具，有空都可以优化；
20. database1.local.xml游戏中有一些代码还会加载这个老的主表，遇到的时候可以修改下；

## 其他

1. 内城的渲染还是基于CCNode之类的代码逻辑，用SortingOrder去模拟的；
2. 很难剥离所有的CCNode代码，只能适配着用；
3. 大地图的核心来自于万国觉醒；
4. 行军集结客户端之前做了优化，只适配当前的逻辑，后期如果加入新的集结可能有很多的不对的地方，要做修改；
5. 由于帝国和红警的代码Code文件夹是同一个外部的SVN地址，所以版本管理的时候要注意版本号；
6. 帝国和红警的代码，在区分的时候，尽量通过容器类去区分，实在不用的，用CCCommonUtils.IsRedAlert()这个接口判断，尽量不要写宏；
7. 每次添加新的管理单例类之后，都要记得在GameController的purgeData函数中添加一下，或者用我的容器类注册；
8. 记得多测试游戏中换账号功能，很多隐藏的BUG都在其中；
9. 自动打包用在测试包，正式包手动管理，避免出错；
10. 游戏中有一些CCB文件还没有删除，是以前cocos2d-x引擎的预制体，好几个还没改，所以一直存在项目中；
11. 客户端的版本管理类是VersionMgr；
12. 所有对项目的优化，都不需要让上层知道，不知道的时候都是自己的时间，知道了之后他还会对你有高要求，同时把这些当成理所当然，就麻烦了；