《王者荣耀》技术总监复盘回炉历程：没跨过这三座大山，就是另一款MOBA霸占市场了

转载

YZF_Kevin 于 2019-09-19 10:17:36 发布 618 收藏 2
分类专栏： 游戏 文章标签： 王者荣耀 王者荣耀技术 王者荣耀技术总监 王者荣耀三座大山
原文链接：https://mp.weixin.qq.com/s/agXeHpN2vkgl5jKsLvi2Ug

编辑导读

2015年8月到10月的两个月间，《王者荣耀》从数据不达标摇身一变成了接下来两年国内最大的爆款。

如今已经大获市场成功的《王者荣耀》一直是业内各方关注的对象，而我们也知道这款产品在成为国民级游戏之前，也遇到过一段鲜有人知的调优期。也就是在2015年8月18号正式不删档测试版本推出之后，被腾讯评级为不达六星之后的时间。

据了解，在8月之后的两个月间，《王者荣耀》技术团队对这个产品进行了非常深度的优化，并攻克了局内同步、网络要求，以及性能表现的三大难关，成功达到了腾讯六星产品的标准。比如延迟、卡顿等不同步问题的出现概率从过去的1%，降低到了0.01%，大幅度地改善了游戏体验。

今天在Unity举办的Unite 2017 开发者大会上，《王者荣耀》项目组技术总监邓君针对这款游戏的调优历程进行了复盘，回顾了这款产品在技术层面遇到的三大问题，以及他们的解决方案。

值得一提的是，参会的开发者也非常关注这次的演讲内容，不仅会长当中人员爆满，围住了整个演讲场地，在演讲结束之后，葡萄君也听到一路上诸多开发者对《王者荣耀》技术经验的探讨。

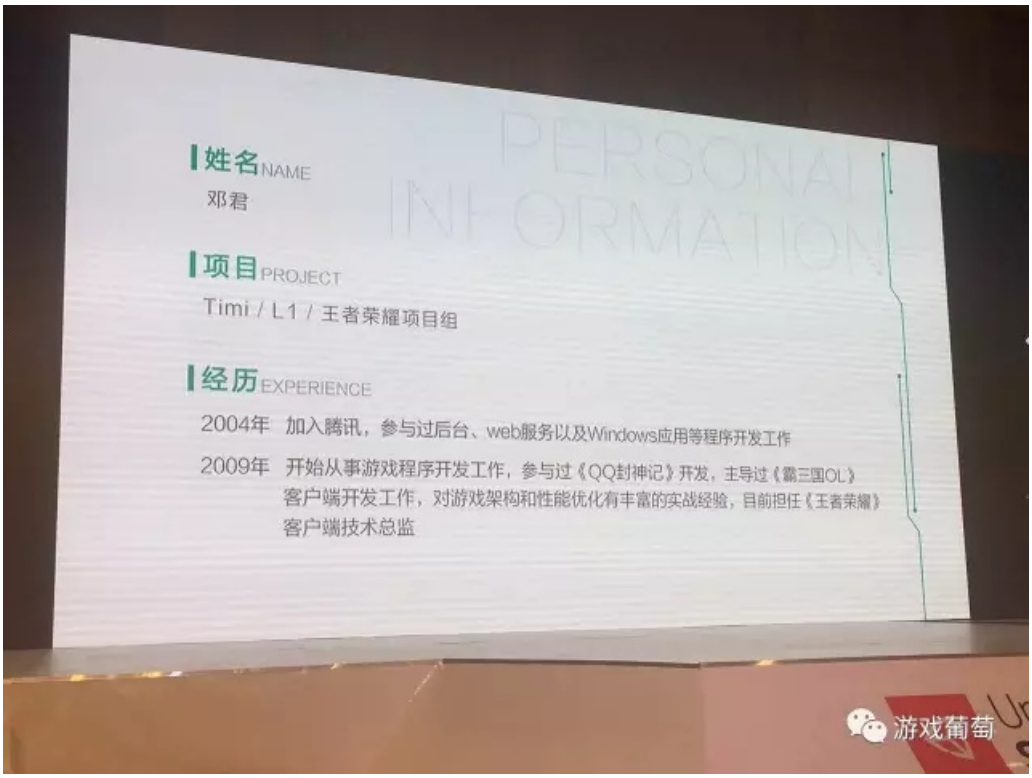
以下内容经游戏葡萄整理发布。

大家好，我是《王者荣耀》的邓君，很高兴今天能够有这样一个机会跟在座的同行一起聊聊技术，互相交流，也感谢Unity提供这样的机会，可以由一个互动。

这次的主题主要是讲一下《王者荣耀》从立项之初经历的惨淡时期到华丽的翻盘，包括碰到技术方面的问题，以及游戏方向上的改变。

我是技术出身的，整个课题也是技术面的，会重点介绍王者荣耀和现在见到大部分不同的技术方案，它实际原理、问题和优化的思路。

先简单自我介绍一下，我是2004年加入腾讯，在腾讯做了4年多的应用层面开发，还包括web各种各样后台都做过，经历比较丰富。在2009年我回到成都，刚好成都的岗位也就只有游戏部门是比较合适的，就转行做游戏了。在成都这边，参与过《QQ封神记》的开发，之后又开发了《霸三国OL》这款游戏，这款游戏开发了三年多，我经历了它从1.0、2.0，再到3.0的版本迭代，这之后才转型做手游，直接做的《王者荣耀》。



现在了解《王者荣耀》或者在玩的人确实比较多，但是我们曾经也没有想过它有这样的结果。当时端游很久都没有做出来成绩，业绩和收入都面临比较大的问题。2012、2011年前后，《霸三国OL》做到1.0版本，游戏中玩家需要控制多个单位，操作起来很难，一开始可以操作5个单位然后变成3个，但即便只有3个单位，操作也让人觉得也很痛苦，于是我们慢慢5个单位的技能合在一个英雄身上，不断地优化。能看到，在MOBA领域，你要做创新，还要脱颖而出，是很难的事情。

在2014年底，2015年初的时候，我们准备组建一个手游团队。因为当时国内市场基本上都在开发手游，能够继续开发端游或者要准备立项端游的非常少，包括腾讯也只有2、3款端游在开发。手游是一个机会，我们当时就希望在2015年把我们的《霸三国OL》端游在手机上呈现。

这个时候我们进行了一个初期Demo的验证，做Demo的只有三个人，引擎、框架、后台，一系列制作下来大概花了两周到三周的时间。这个Demo里有基本的进游戏、选人，然后可以释放技能，正常的战斗，到结算。

Demo的引擎我们采用了Unity，做完之后也觉得Unity很好用，开发效率确实比较高。

2014年年底的时候，我们制作人去公司开会，当时做一个非常明智的决策：我们需要马上暂停端游的开发，直接做手游。就是这样的一次决策，真正地扭转了我们整个团队的命运。如果晚一年，可能今天爆红的MOBA游戏就是另外一个，而不是王者了。

于是接下来在2015年，我们才有了想法开始独立招聘20、30人来做这个手游项目。

从端游转型做手游，肯定要面临选择，到底要用什么样的引擎，采用什么样的方案进行手游的开发。当时，腾讯以及成都周边的创业团队，基本上都用Unity，我们做Demo的时候，也选择大家用过的，已经有产品进行验证的引擎，同时我们也考察它适不适合我们的团队。

Unity在我们当时做Demo时的理解来看，它确实对中小团队，甚至对一些大型项目来说，都有几个比较明显的优势。

1.易上手，我们花三周就可以做出Demo，可以看到易上手是它的一个非常大的优势。

2.它的工具都是很完善的，能够做到一站式解决，你不需要在这里面下载工具，那里面额外补充一些插件。

3.它插件资源很丰富，我们从最开始做Demo的时候，基本上都可以从Asset Store中找到一些可以用来验证我们想法的资源，可以加快我们开发的效率。

4.上面这三点加起来，就能体现出它非常明显的优势，即开发效率特别高。

5.它还有跨平台的优势，它本就是跨平台的引擎。

6.最后它还能让你更方便地补充技术人员，社招也很容易招聘到熟悉Unity的开发人员。

对比以前我们自己做引擎，或者用过其他的引擎，从效率上来讲，最终我们选择了一个开发效率最高的引擎Unity。

我们从端游转手游是在2014年底，但真正开始研发《王者荣耀》是在2015年3月份的时候，这个时候项目的要求是让开发的周期尽量短，尽快把手游做上线。我们原本在《霸三国OL》的开发团队大概有40、50个人，再加上后来加入的人员，形成了100多人的团队，进行了游戏的开发。

在2013、2014年的时候，手游在PvP的方面都比较弱，大部分是卡牌游戏、单机游戏。我们原本是做的端游，它的生命力包括趣味性也是很足的，所以我们做手游的目标，就是即使游戏里会存在PvE的闯关内容，但我们的核心还是要把PvP做好，让玩家有真正的对抗，让玩家与玩家有交流，能体会到这样的游戏乐趣。

既然选择PvP，那么这款产品就是一个网络游戏，而选择网络游戏的同步机制，就是我们首要考虑的问题。同步机制中最常见的应该是CS状态同步，我们的端游也是这样做的，当然，状态同步也有他的优缺点。

先看一下状态同步的优点。

第一，它的安全性非常高，外挂基本上没有什么能力从中收益。

第二，状态同步对于网络的带宽和抖动包有更强的适应能力，即便出现了200、300的输入延迟再恢复正常，玩家其实也感受不到不太舒服的地方。

第三，在开发游戏过程中，它的断线重连比较快，如果我的游戏崩溃了，客户端重启之后只需要服务器把所有重要对象的状态再同步一次过来，重新再创建出来就可以了。

第四，它的客户端性能优化优势也比较明显，比如优化时可以做裁剪，玩家看不到的角色可以不用创建，不用对它进行运算，节省消耗。

再说一下我认为的缺点。

第一，它的开发效率相对帧同步而言要差一些，很多时候你需要保证服务器与客户端的每一个角色对象的状态之间保持一致，但事实上你很难做到一致。

比如客户端和服务器端更新的频率，对优化的一些裁剪，网络的抖动等等，你要让每一个状态在客户端同步是比较难的，而你要想调试这些东西，来优化它带来的漏洞、不一致的现象，花费的周期也会比较长，想要达到优化好的水平也比较难。

第二，它比较难做出动作类游戏打击感和精确性。比如说你要做一个射击类角色，他的子弹每秒钟要产生几十颗，基于状态同步来做是比较难的，因为系统在很短时间内，会产生很多数据，要通过创建、销毁、位置运算来同步。

第三，它的流量会随着游戏的复杂度，而逐渐增长，比如角色的多少。我们做《王者荣耀》时，希望在3G、4G的网络条件下也能够玩PvP，所以我们希望它对付费流量的消耗能控制在比较合理的水平，不希望打一局游戏就消耗几十兆的数据流量。

而另一种同步策略是帧同步。

这种技术应用的很广泛，最早的《星际争霸》《魔兽争霸3》都采用了帧同步，他们都基于局域网运行，网络的条件非常好，也不需要服务器就能搞定。帧同步的优点有几个：

第一，它的开发效率比较高。如果你开发思路的整体框架是验证可行的，如果你把它的缺点解决了，那么你的开发思路完全就跟写单机一样，你只需要遵从这样的思路，尽量保证性能，程序该怎么写就怎么写。

比如我们以前要在状态同步下面做一个复杂的技能，有很多段位的技能，可能要开发好几天，才能有一个稍微过得去的结果，而在帧同步下面，英雄做多段位技能很可能半天就搞定了。

第二，它能实现更强的打击感，打击感强除了我们说的各种反馈、特效、音效外，还有它的准确性。利用帧同步，游戏里面看到这些挥舞的动作，就能做到在比较准确的时刻产生反馈，以及动作本身的密度也可以做到很高的频率，这在状态同步下是比较难做的。

第三，它的流量消耗是稳定的。大家应该看过《星级争霸》的录像，它只有几百K的大小，这里面只有驱动游戏的输入序列。帧同步只会随着玩家数量的增多，流量才会增长，如果玩家数量固定的话，不管你的游戏有多复杂，你的角色有多少，流量消耗基本上都是稳定的。这点延伸开来还有一个好处，就是可以更方便地实现观战，录像的存储、回放，以及基于录像文件的后续处理。

帧同步也有它的缺点。

第一，最致命的缺点是网络要求比较高，帧同步是锁帧的，如果有网络的抖动，一段时间调用次数不稳定，网络命令的延迟就会挤压，引起卡顿。

第二，它的反外挂能力很弱，帧同步的逻辑都在客户端里面，你可以比较容易的修改它。但为什么《王者荣耀》敢用帧同步，一方面是因为当时立项的时候开发周期很短，半年时间要做上线，要有几十个英雄，存在时间的压力，另一方面，MOBA类游戏不像数值成长类的游戏，它的玩法是基于单局的，单局的作弊修改，顶多影响这一局的胜负，不会存档，不会出现刷多少钱刷多少好的装备的问题，而且作弊之后我们也很容易监测到，并给予应有的惩罚，所以我们认为这不是致命的缺点。

第三，它的断线重回时间很长，相信台下也有很多王者玩家，也曾碰到过闪退以后重回加载非常长的情况，甚至加载完以后游戏也快结束了，这是帧同步比较致命的问题。

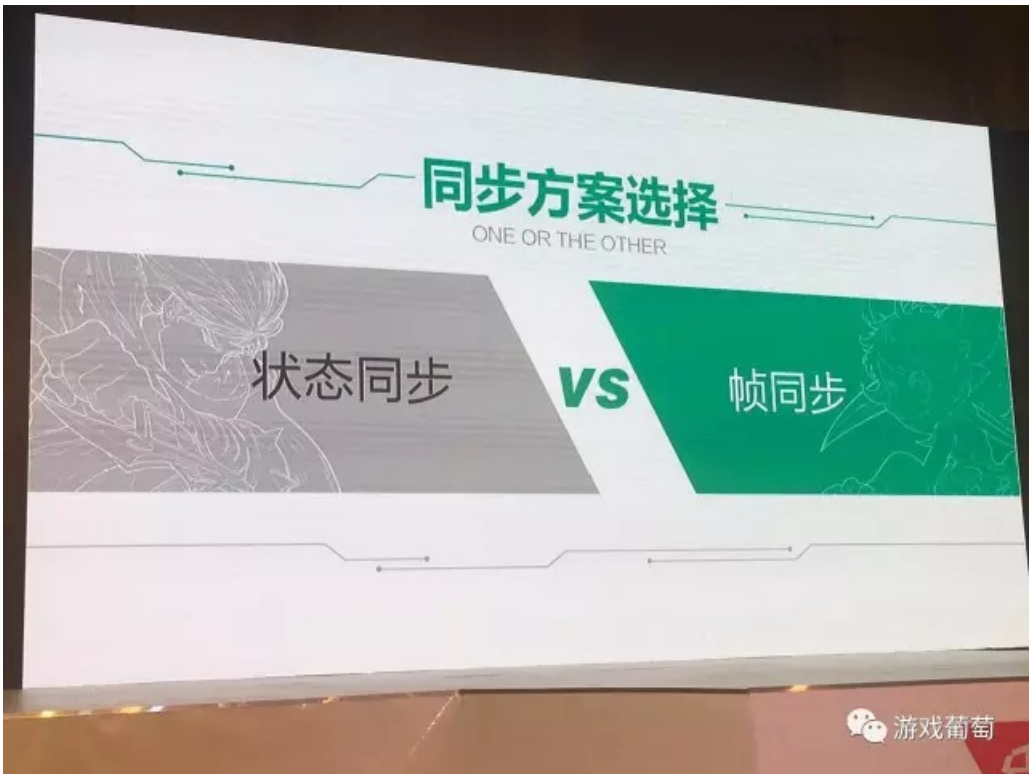
第四，它的逻辑性能优化有很大的压力。大家应该没有见到哪一款大型游戏是用帧同步来做的，因为这些游戏的每一个逻辑对象都是需要在客户端进行运算的。如果你做一个主城，主城里面有上千人，上千人虽然玩家看不到它，但游戏仍然需要对他们进行有效的逻辑运算，所以帧同步无法做非常多的对象都需要更新的游戏场景。

那么我们为什么选择了帧同步而放弃了状态同步呢？

我们前面提到它两个优点缺点是相对的，这边的优点对于那边来说就是缺点。对于我们手游立项的时候，最重要就是时间。当时市面上正在开发的MOBA手游不止王者一款，大家都在争上线的时间，所以我们要选择一个开发周期最短的方案。

然后我们做端游的时候也有一个深刻的体会，如果要做有趣的英雄，有趣的技能，它在状态同步上面很难调出一个比较满意的效果。所以最后我们依然选择帧同步的方案。

现在来看，选择帧同步方案之后，我们再把它的缺点进行优化或是规避，之后它带来的好处是非常明显的。《王者荣耀》重除了英雄的设计以及技能的感觉，还有很重要的一点，就是它确实在做一些非常有特色的英雄，它的技能、反馈、体验上面都做得不错，这些都是基于帧同步技术方案带来的优势。



我们选择了方案之后，当时觉得很high，觉得这样一个技术方案开发起来得心应手，效率如此之高，做出来的效果也很好。

但事实上，它也有好的一面，也有坏的一面，技术测试版本上线后质量不好，其中技术层面遇到的问题就是下面这三座大山。

第一是同步性，同步性这块容易解决，其实也解决了；

第二也是最大一块网络问题，帧同步它的网络问题导致我们对它技术方案的原理没有吃透，碰到了一些问题，那时候游戏的延迟很重，画面卡顿，能明显感觉走路抖动的现象；

第三是性能问题，这个问题始终存在，我们也一直在优化。

第一座大山，最容易解决的同步问题。

帧同步的技术原理相当简单，10、20年前在应用这种技术了，从一个相同初始的状态开始，获得一个相同的输入，往下一帧一帧执行，执行时所有代码的流程走得都是一样的，这个结果调用完了以后，又有一个新状态，完成循环。相同的状态，相同的流程，不停的这样循环下去。

这个原理虽然简单，但是你要去实现它的时候，还是会有很多坑。

右边写的是实现要点，这是我们在解决第一座大山经验的总结，也是我们实际开发过程当中做的事情。

首先，我们所有的运算都是基于整数，没有浮点数。浮点数是用分子分母表达的。

其次，我们还会用到第三方的组件，帧组件也要需要进行一个比较严格的甄别。我们本身用的公司里面关于时间轴的编辑器里面，最初也是是浮点数，我们都是进行重写改造的。

再次，很多人初次接触帧同步里面的问题，就是在写逻辑的时候和本地进行了关联、和“我”相关，这样就导致不同客户端走到了不同的分支。实际上，真正客户端跟逻辑的话，要跟我这样一个概念无关。

接下来还有随机数，这个要严格一致。这是实现的要点，严格按照这上面的规则写代码还是有可能不同步，本身就很难杜绝这样的问题。

最后，真正重要的是开发者要提升自己发现不同步问题的能力，什么时候不同步了，不同步你还要知道不同步在什么点，这是最关键的。你需要通过你的经验和总结提升这样的能力。这个能力还是通过输出来看不同客户端不同输出，找到发生在什么点。

比如在《王者荣耀》里，我们看到不同步的现象应该是这样，有人对着墙跑，你看到的和别人玩的游戏是不一样的，就像进入平行世界。

最开始测试《王者荣耀》的，我们希望不同步率达到1%，就是100局里面有1局出现不同步，我们就算游戏合格，但实际上对于这么大体量游戏来说，这个比率是有问题的，经过我们不停的努力，现在已经控制在万分之几，一万局游戏里面，可能有几局是不同步的。

这个问题不一定是代码原因或者没有遵循这些要点才出现的，有可能是你去修改内存，你去加载资源的时候，本地资源有损害或者缺失，或者是异常。说白了，你没有办法往下执行，大家走了不同分支，这都可能引起最终是不同步的。

如果你不同步概率比较低，到了这种万分之几概率的时候，很难通过测试来去还原，去找到这样不同步的点。

最开始我们游戏出现不同步的时候，就是在周末玩家开黑多的时候，随着你的概率越来越低，基本上你就自己就还原不出这些问题了，只能依靠玩家帮你还原这样的场景，来分析这样的不同步问题。

同步性遵循这样的要点，按照这样的思路来写，加上你不同步定位的能力，有了监控手段能够去发现，这个问题其实就解决了。解决之后，你就可以好好享受帧同步的开发优势。

第二座大山就是网络，《王者荣耀》技术测试版本出台的时候，延迟非常大，而且还是卡顿，现在看一下帧同步里面比较特别的地方。帧同步有点像在看电影，它传统的帧同步需要有buffer，每个玩家输入会转发给所有客户端，互相会有编号，按顺序输入帧。

比如我现在已经收到第N帧，只有当我收到第N+1帧的时候，第N这一帧我才可以执行。服务器会按照一定的频率，不同的给大家同步帧编号，包括这一帧的输入带给客户端，如果带一帧给你的数据你拿到之后就执行，下一帧数据没来就不能执行，它的结果就是卡顿。

网络绝对理想的情况下还好，但现实的网络环境不是这样的。帧同步要解决问题就是调试buffer，以前有动态的buffer，它有1到n这样的缓冲区，根据网络抖动的情况，收入然后放到队列里面。

这个buffer的大小，会影响到延迟和卡顿。如果你的buffer越小，你的延迟就越低，你拿到以后你不需要缓冲等待，马上就可以执行。但是如果下一帧没来，buffer很小，你就不能执行，最终导致的结果你的延迟还好，但是卡顿很明显。

如果调到帧同步的buffer，假如我们认为网络延迟是1秒，你抖动调到1秒，那得到的结果虽然你画面不抖动了，但是你的延迟极其高。如果连最坏的网络情况都考虑进去，buffer足够大，那么记过就跟看视频是一样的，平行的东西，看你调大条小。一些局部的措施我们都做过，都是一样的问题。

具体我们怎么优化卡顿的问题呢？

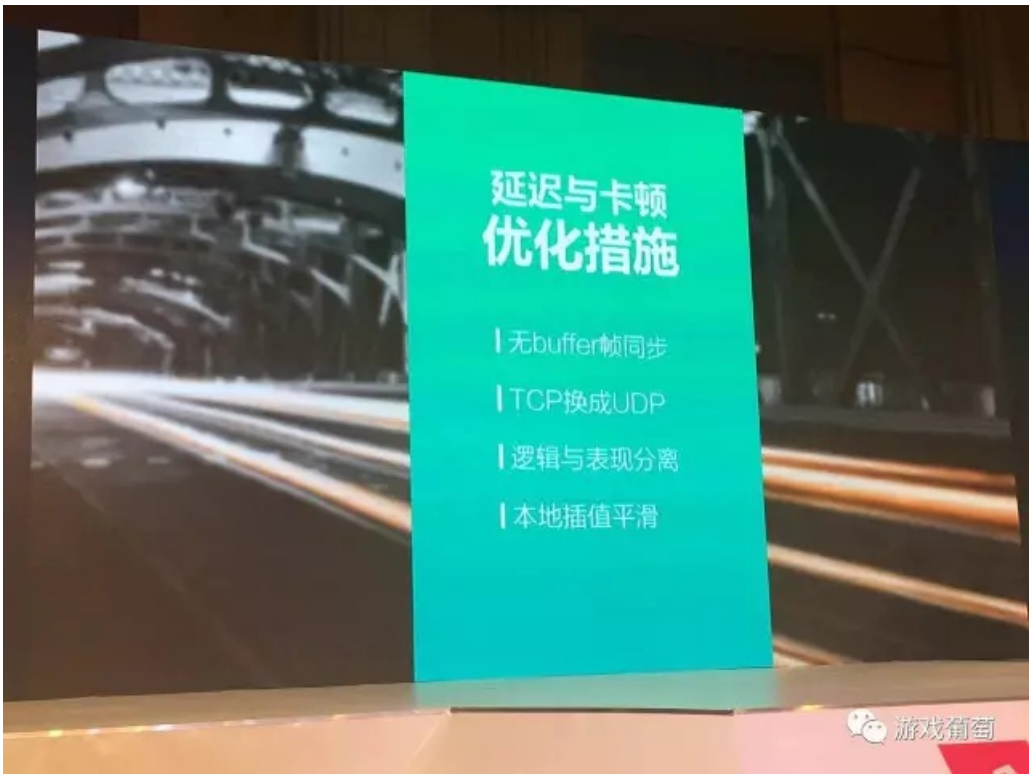
刚才提到该帧同步与buffer，这个buffer可以是1也可以到n，我们要解决我们的延迟问题，我们就让buffer足够小。事实上《王者荣耀》最后做到的buffer是零，它不需要buffer，服务器给了我n，马上知道是n，我收到n，我知道下一次肯定是n+1，所以我收到n之后马上就把n这一帧的输入执行了。

那么为什么不卡顿了，画面不抖动了？

最后一个关键点，是本地插值平滑加逻辑与表现分离。客户端只负责一些模型、动画、它的位置，它会根据绑定的逻辑对象状态、速度、方向来进行一个插值，这样可以做到我们的逻辑帧率和渲染帧率不一样，但是做了插值平滑和逻辑表现分离，画面不抖了，延迟感也是很好的。

做了这些后，我们还把TCP换成UDP，在手机环境下，弱网的情况下，TCP很难恢复重连，所以最后用了UDP来做。整体来说，在网络好的情况下，它延迟也是很好的，在网络比较差的情况下做插值，也是传统CS的表现。

我们经常见到角色A和B，有些客户端A在左B在右，有些是A在右B在左，帧同步逻辑上面AB之间的距离和坐标都是完全一样，但是画面上看到他们可能会不重合，那就是你把它们分离之后的表现。网络极其好的情况下，它应该是重合的，但是在网络差的情况下，可能会有些偏差。这里面是最重要的一块优化。



第三座大山，是我们对性能的优化。

本身帧同步逻辑上面在优化上面存在一些缺点，所有的角色都需要进行运算。这方面我们也是借助Unity的特性，如果你想追求性能上的极致，有些东西你需要寻求好的方式。

第一点是热点的处理。

我们是不用反射的，它都有GC性能开销，我们的做法里面，会把对象的显示隐藏放在不同的渲染层里面，尽量让整个游戏帧率是平滑的过程。还有我们本身有自己的系统，比如AI，在《王者荣耀》这样的多角色游戏中，你如果想要做出比较好的体验，那么AI就要做得比较复杂。

而要去优化热点，我觉得就只有这三个步骤可以走。

首先，从程序的结构上面能找到更优的，它的优化效果就是最明显的；其次，如果你的结构都是用的最好，就去挖掘局部的算法，调整你代码的一些写法。最后，如果局部的算法都已经调到最优还是没有什么办法，那只有一条路，就是牺牲整个质量，就是分帧降频。

第二点是GC，这块刚才说不用反射，还有装箱和拆箱的行为也是尽量少用。Unity指导过我们的优化，从GC上面的考虑，他们建议每一帧应该在200个字节以内是比较好的状态，其实很难做到，王者也是每一帧在1k左右，很难做到200。

第三点是Drawcall，这些传统的优化手段大家都用的很熟了。

第四点是裁剪，帧同步里面是不能裁剪的，表现里面我看不到的可以降低频率或者不更新它，这在表现里面可以做的。

第五点是3DUI的优化，比如《王者荣耀》的血条、小地图上面叠的元素等等，这些UI都比较丰富，这块我们用了3UI的方式来优化，没有用UGUI里面进行血条方面的处理。

我们也牺牲了一些东西，我们把所有东西都加载了，在游戏过程当中，我们希望不要有任何IO行为，包括输出我们都是要布局的。你处理的决策和复杂度，如果在一帧里面放出100颗子弹，在放100颗子弹的时候一定要掉帧的，一定要在力所能及的时候把这些东西做到极致。

上面提的是我们的第一代，也是在去年5月份以前做的优化方案。5月份以后，我们还做了另外一件事情：GameCore。

首先，为什么我们觉得iOS比安卓的优化效率高一些，一方面是iOS的CPU架构包括系统确实都优化的比较好，另一方面我们用的Unity4.6，在IOS下面它本身效率高一些，在安卓端的机器各种各样，性能也是千差万别，我们只能用性能比较差的方式。

因为我们已经做到逻辑和表现的分离，那么我们能不能把逻辑独立出来，做成一个C++的东西，实际上我们在去年开始已经在这样做了。做之前也测试过C++和Mono性能的差别，大概是2.5左右，本身我们的逻辑占比游戏消耗20%多，逻辑不是一个大头，我们做了这件事情之后，还是有效的，帧率提升了2到3帧，花的时间很长。

其次，做GameCore以后最明显的变化是我们以前逻辑上的GC没有了，我们有自己内存的管理、对象的管理，包括里面所有的容器类这些东西都是我们自己实现的，包括反射整个一套。它有了自己的内存管理，本身的效率就会比较高，这就已经是一个比较明显的优势了。

再次，有了GameCore之后，又多了很多应用场景，这个东西就是玩法的服务器版本，应用场景运行服务器要做很多的分析，还有第三方使用都是可以的。

最后，GameCore还有可以扩展多线程的潜力。

今后，我们也有几个计划。

第一，我们考虑能不能在热更新上面有所突破。因为王者这样一个游戏类型，包括它的体量，我们对于性能有一个比较极致的追求，不会轻易使用脚本层面在性能层面本身就不是最好的。这个我们要去研究的就是热更新，性能最好的方式。

第二，我们也在和硬件厂商沟通，他们希望游戏能够真正发挥多核性能上的优势，大部分的游戏在单核上面，把一个核吃的满满的，很多时候我们现在得出的结论，GPU性能也很强，王者并没有对GPU占满，可能只用了30%，CPU反而吃的比较满，吃满以后它还有另外一个坏处，它的发热、降频，你如果用多线程、多核去尽量平坦，让它不要处于高频的工作方式，反而会有更好的效果。

第三，我们现在用的是Unity 4.6版本，但Unity已经进化到5.7版了，后面他们还会推出新的特性，我们希望结合一些Unity新特性，现在已经有些游戏用5.6可以提升性能。

最后，不光是提升性能问题，Unity在多线程的渲染，也有很好的作用，使用引擎优势也是很必要的。随着性能的提升，我们会对王者的画质进行升级。

好的，我今天的演讲就到此结束了。