Flash11 Stage3D游戏《封神无双》制作团队专访

转载

dj0379 于 2012-11-15 10:57:26 发布 3678 收藏
分类专栏： AS3FlexFlash


AS3FlexFlash 专栏收录该内容
22 篇文章 0 订阅
订阅专栏

《封神无双》介绍

《封神无双》是首款基于Flash11 Stage3D技术的3D MMORPG网页游戏。已于今年8月底正式内测，运营商是昆仑万维，研发公司为广州第九艺术网络科技。目前游戏处于调整期并在持续添加游戏玩法，同时进一步改善游戏性能，争取多方位提升游戏可玩性。

公司起源

广州九艺成立于2007年底，核心成员均从业10年以上传统客户端游戏开发，公司成立时产品和定位主要以PC客户端游戏为主，第一款2.5D MMORPG产品《古域》已由畅游代理运营至今。由于2010年后网页游戏势头迅猛，于是我们也于2011年初开始尝试网页游戏，这也就是封神无双项目的契机。

开发过程中5件对的事情

1、选择Flash11作为解决方案

主要因为Stage3D使开发web平台上基于硬件渲染的3D游戏成为可能。在2011年项目调研期，我们还在犹豫开发2D游戏还是3D游戏，虽然2D页游技术门槛低，风险小，但由于我们初涉页游领域，经验不足，在成熟的2D市场上难有竞争力，玩法也很难创新。相反3D虽然技术难度大，风险高，但是一种技术创新，熟悉传统3D开发的我们也有明显优势，而且我个人从技术人员角度更想开发3D游戏，在无插件3D页游方向，我甚至还为此研究过webGL，不过那时也还刚起步，现在都还没成熟（html5)。适逢stage3D第一版本molehill在那时推出，得知此消息后我们立即开始调研， 经过初步分析，有较大风险，但我们主要看中flash的普及率和浏览器集成 以及adobe公司的技术实力，最终还是选择用flash11来开发页游，选用一个还在孵化阶段的技术用于正式商业项目，在当时的确是一个很大胆的做法。不过随着项目基本成型后，并且Adobe有条不紊的跟进和更新最后release，这个风险也基本消除了，验证了我们当初技术选型的正确性，主要抓住了“无插件3D页游“这一先机。

2、 中期抛弃效率不甚理想的Away3D引擎，重写渲染核心，大幅优化效率

初期调研期间，为了快速开发原型已验证可行性，我选用了开源的Away3D作为渲染模块，包含在我们自己新命名的引擎DelatX当中。开源引擎是当时的最优选择，因为flash3D都是大家都是刚刚起跑，都是在尝试，问题也很多，能跟踪源码自己修改问题比使用一个不稳定的黑箱要安心很多。当时只推出了alpha版，bug很多，后来因为改动较大，我们也不再合并away3d更新。本来打算在它基础上继续开发，越来越发现效率实在跟不上，主要是渲染部分，很多基础算法就不是优化算法（比如骨骼动画、材质系统、shader集成），它想做大而全的图形开发包，但架构偏复杂，中间层过多，相对于我们端游的引擎的架构过于笨重过于缓慢。于是我们几乎抛弃了所有away3d的代码，只保留了一些纯粹的辅助包，基于简洁、高效的原则将整个渲染模块重写，摒弃不需要的，只留需要的代码，最后性能有大幅度提升。整块优化和重构工作（包括下面提到的另外3件正确的事情）主要是由我们的CTO柯达招操刀，我协助完成。

3、 采用Stage3D实现界面系统

项目初期，界面系统我是直接采用当时开源的aswing界面库，然后做一定扩展以兼容我们的界面系统（从端游移植而来）。初衷是想减少开发成本，但aswing也是相当庞大和复杂的，这个兼容过程也耗费了相当大的人力。后来由于目前stage3D的内容不能绘制在2D的flash元件之上，导致有些需求基于2D的界面无法实现，比如需要在界面上绘制3D模型和特效，后来考虑再三，决定除了主画面，界面也用stage3D来实现绘制，和3D的绘制统一起来，在项目中后期实施这个改动也是一大手术，整个过程持续接近一个月，但结果证明是正确的，不仅实现了需求，由于绘制转移到了GPU，CPU大幅下降，并且内存也降低了不少。

4、使用自定义ds格式shader解释器，提高shader开发效率

这个其实也属于渲染引擎重构和优化的一部分。因为我们发现书写AGAL的shader效率很低，调试也困难，而当时尝试过的pixelBender3D也有很多bug，几乎不能使用。汇编的书写主要麻烦在与寄存器的分配，比如mov vt0,v1; mov vt1, v2; 假如去掉了v1属性，那么需要修改所有v2为v1，临时寄存器的修改更是会影响到整段shader代码的调整。能否让

我们不关心那些寄存器编号，只关注我们需要的变量和操作呢？比如mov temp, pos; mov anotherTemp, color; 基于这个原则，我们开发了一个轻量的解释器，输入刚才那段有具名变量的伪shader，指令集完全沿用AGAL，输出的则是原生的AGAL代码，寄存器全部由解释器分配好，无需人工干预。这大大简化了shader的开发，写shader不比高级语言麻烦多少，而且直接面对底层指令，反而更容易发现问题，对于复杂的计算，只要旁边附上高级语言的注释就可以了。目前这个解释器已支持基本替换变量、语义绑定以及函数（宏），以后还想添加的特性比如文件的include、寄存器按分量更精确分配等。

5、特效系统核心移植到GPU用shader实现

ActionScript3虽然在脚本语言中效率不算低，但终究还是脚本语言，性能难以和C/C++匹敌，尤其是CPU密集型计算更是差别巨大，导致原本我们在端游的特效系统移植到flash后有很大的性能瓶颈，所以做了很多制作上的限制。在项目后期，外界测试反馈的效率问题比较严重的情况下，我们不得不进行特效系统的优化，主要方案是把性能瓶颈的特效更新代码移植到GPU用shader去实现，as端只传输基本的数据和参数，不做多余的计算。这部分优化也让特效的限制基本解除，效率和效果都有显著提升。

开发过程中的5个问题

1、性能需要进一步优化

虽然我们一直强调性能，并且胜于画面（首先要让玩家能玩），也投入了很多时间在优化上，但似乎已经到了一个瓶颈，受限于flash虚拟机本身，所以还没有完全达到预期的结果。另外我们有一个比较隐晦的内存泄露也还在排查当中，目前并没有发现太好的内存分析的工具，包括能分析flash内部对象如BitmapData、String等对象内存。Monocle是一个很好的性能分析工具，据说新版本会添加内存分析功能，值得一试。

2、逻辑代码的质量审核需要加强

《封神无双》在画面上做出了一定水准，效率也不错，但这只是游戏的基础，游戏的主体功能目前还存在不少bug，影响到用户体验，也会导致玩家流失。我们团队规模较小，很多流程也不太正规，开发节奏比较快，导致代码质量的把关上出现了脱节，这也是日后需要重点改善的地方，我们需要的是更高质量的产品。

3、加强团队内部的培训

这其实和上一条有关联，在新人培训上我们还停留在传帮带和放在一边自己啃的层次，很多东西包括项目的系统讲解、常规开发事项等都应该在前期开始就贯彻执行，降低开发期间的重复工作，提高开发效率。

4、尽可能带动解决flash11的兼容性问题

flash11虽然带来了技术的变革，把3D搬到了网页，但变革的路都是曲折的。比如低端显卡兼容性、旧驱动的兼容性，右键开放后和浏览器自带手势的冲突问题，chrome浏览器安装多个版本flash插件的冲突，这个bug甚至会引起stage3D回退到软件渲染，还有其他一些影响玩家进入的问题。毕竟我们是第一个吃螃蟹的，从用户那里得到了很多关于flash插件的问题，也积极反馈给adobe以求解决，但这是一个漫长的过程，随着这些问题的逐个解决，我们才能真正说flash3D页游普及到90%的PC端以上。

5、处理用户反馈的速度

这方面我们还远远不够，这需要在各方面下功夫，人力问题、个人解决问题能力提升、游戏更新方式的优化等，比如做通过后台入口做游戏的线上修改，迅速地处理问题，这是玩家和运营商都喜闻乐见的。

技术与效率

选择Adobe技术理由是跨平台，开发效率高，容易学习。作为网页游戏的主打平台flash的开发商，没有理由不去选择。

目前在使用Adobe的解决方案有：

i. Flash Player 11.4, Stage3D
ii. Flash Builder 4.5
iii. Monocle

如何实施Adobe解决方案

游戏的实施和传统网络游戏开发没有区别，策划定好世界观，做系统的设计，提出明确的需求交给美术和程序去实施。美术部分主要是用3dsmax、Photoshop/Coraldraw/painter等传统艺术工具进行基本素材编辑，然后导入自研的美术工具集加工处理成支持我们游戏引擎的资源格式，最后由程序调用并呈现在游戏当中。游戏区别于其他页游主要还是在基于flash硬件加速的3D游戏这一技术创新点上。

Adobe解决方案（AIR, Flash, Stage 3D等）本身的独特优势

Stage3D 高性能的跨平台3D图形API极大的提升异于传统页游的表现力，接近客户端游戏效果，借助flash本身的高占有率，这一优势会让其他跨平台3D技术短期很难超越，flash平台也被Unity、Unreal等游戏引擎商看中，flash导出也成一趋势，因为潜在的用户是非常可观的。

Adobe解决方案带来的项目收益

开发效率相对传统端游C++开发有大幅提升，主要体现在编译时间的大幅降低，以及语言和平台本身的易用性。Monocle的使用也大大减少查效率问题的时间，因为它能非常快并准确的定位到效率瓶颈之处。

分享该游戏的制作经验
由于初次接触页游开发，遇到许多flash传统开发的常见问题，也适逢flash11的各种变革，一方面我们要虚心向传统flash开发人请教学习开发心得，一方面在要积极跟进adobe，及时反馈，及时采用新的方案和工具（比如monocle），减少弯路。

性能永远优于画面。页游用户对画面要求并不如端游高，所以对画面的追求要有个度，玩家的体验主要还是看性能和流畅度。第一是下载效率和加载效率， 最快的让玩家进入游戏，资源量和资源大小的控制要达到极致才行，我们已经做过很多努力，贴图的jpegXR压缩，文件的swf压缩，并行的下载队列等等。第二是游戏运行效率，包括渲染效率和操作流畅度，因为CPU方面是瓶颈，GPU方面性能并不是优化重头，提高as代码执行效率，降低CPU占用是关键，借助性能分析工具也能达到一定高度。

要明确区分页游和端游的开发。要认清他们的本质区别，而不仅仅是把端游或者主机游戏移植到网页端，除了减少下载外并无太多意义。页游注重休闲和零碎时间的利用，注重用户体验，并尽可能的降低用户操作，所以如果你是要移植已有的端游，在游戏系统得做大量调整，UI/UE也要做很多优化。

团队目前进行的工作以及未来的项目

目前我们着眼于性能的进一步提升，以及游戏玩法的丰富和游戏bug的修复。另外公司正着手进行基于ActionScript3和C++结合的游戏微端开发，基于avm2开源虚拟机和底层C++ DirectX图形引擎，以充分利用ActionScript的高开发效率以及C++的运行效率。未来可能会尝试移动平台的游戏研发。