Bioshock iOS Postmortem

2013.3-2014.8

03.11.2013 Kick off

05.08.2013 New Havok pass compile

08.08.2013 Add some new Havok animation

08.09.2013 Combine ES2 codes, game can run on PC with incorrect graphics

08.26.2013 Add iOS codes

09.20.2013 Complete deploying app to device through build tool on PC

09.26.2013 Game can run on device with simple control (static mesh, BSP, no animation)

10.11.2013 Run test map on device(with texture, animation)

10.24.2013 Add particle on device

10.28.2013 Complete uncompressed data making

11.15.2013 Complete 1st version control: include control button - plasmid & fire

8.13.2014 Submit

# Planning

项目定义是否清晰

前期工作是否充分

是否有足够的时间来做计划

是否以及如何较准确地预估项目所需的时间和难度

是否预留足够的缓冲区

如何评估方案的可行性

## What we did right

制定了计划，呵呵

## What we did wrong

前期计划制定比较随意，没有一起开讨论会

制定计划比较仓促，没有足够的前期研究，导致后面计划和预估不准

对某些问题的初期研究不充分，计划中没有预留足够的缓冲区，导致后面时间不够。比如Havok的移植比预期多了很多时间，最终导致从项目初期就开始加班和项目延期。

后来图形的实现也跟原来的计划不一样，花费的时候也比预估的多

我们拿到的原始移植版本不是最终版，本身有很多bug

## Improvement/Suggestion

制定计划时项目组成员一起开个讨论会

应该有足够的时间来做研究，然后再制定计划和确定schedule。比如先让2，3个人研究一个月左右再制定计划。

计划制定时要预留足够的缓冲期

对于移植项目，原始版本比较重要，应该是最终版本

# Porting

Porting过程中的主要困难是什么：数据（内存对齐，特别是含有位域的struct的对齐）; Math function（abs）

移动平台项目的主要问题是什么

## What we did right

## What we did wrong

build系统不是很完善，经常遇到不同人编出来的版本都不一样的情况，有些bug自己测试没问题但会被QA打回来，这样还会重复好几次

iOS版跑起来的时间有点晚

有一些冗余代码没有去掉：比如Bink，SWF模块的一些代码

## Improvement/Suggestion

Porting过程中的主要困难是：数据（内存对齐，特别是含有位域的struct的对齐）; Math function（abs）

应该尽快让版本在发售平台上跑起来，然后再该平台上做后期开发

内存和性能是iOS移植/开发的两个重要问题

# Implementing

任务分配是否明确清晰

每个人分工是否合理有效

如何将需求分割成若干个任务：UI

任务优先级划分

是否因为依赖其他人的工作完成结果而导致任务等待

是否需要维护项目文档以及方式

功能共享

哪种工作方式更有效：大块任务or任务细分，单人or合作，push or自我管理

UI

## What we did right

每个人都有较明确的分工：UI，Graphics， Animation， Physics， GamePlay

## What we did wrong

每个人只专注于自己的一块，对其他人的工作或者其它模块不是很清楚，导致有人请假或者离职对项目的影响会比较大

Havok遇到很多预先没有估计到的问题，严重地拖延了项目进度

资源修改后常常没有在游戏里面确认过就上传了

没有做技术共享，每个人做的东西只有自己才知道，即使写了文档也很凌乱。

很多人提交p4v时对于提交的描述比较随意

## Improvement/Suggestion

对于有依赖关系的任务，需要分优先级。

对于任务分配，不用分配得太细，可以分配大块的任务给某个人，然后每个人自己细分或者讨论决定。

对于统一类型的大块任务，最好至少有2个人做，降低因为人员流失或者突发事件导致的风险。

遇到问题可以结对编程，效果会比较好。

所有对资源的修改都需要在设备上验证没有错误后才能上传，因为很多资源有问题很难查，很可能要花几天才能查到是资源问题。这需要美术leader和Producer来制定check制度。

需要建立技术共享平台或标准：比如建立类似wiki的网站，可以随时上传一些需要分享的东西，也便于积累和整合。

每个阶段（比如大的milestone或者某块功能做完后）都做一下总结，做下演示或者讲解：比如UI可以演示一下做一个简单的screen的步骤；做完Animation可以开一个讲解会，介绍下游戏中用到的Animation系统以及演示下动画数据制作的流程。

要做code review。

对于p4v的提交，描述一定要详细，描述不是个自己看的而是给别人看的。

是否能让IT做一个版本管理与bug report的关联系统，比如p4v与TTP关联，这个非常有用，不过可能难度比较大。

# Assets

资源是否齐备

资源维护是否合理

## What we did right

## What we did wrong

移植的original版本不是最终版，本身有很多bug

后期出现很多资源不对的情况，要找出错误，重新做资源，重做版本，很费时。

## Improvement/Suggestion

美术资源修改后要装到设备上跑游戏验证下，确定没问题后再上传。

项目中后期要每天做测试保证资源数据没问题

# Management

项目风险控制

项目进度管理

版本维护

代码Review的成本及必要性

如何提高项目过程中的技术分享

任务分配是否合理、有效

对自己分配到的任务的看法，是否乐于完成所分配的任务

专才、全才？

是否将太多的时间用于开会，太多/太少？

每个人是否能感受到自己的价值

角色定位是否清晰

如何发挥自己的特长

加班及效率问题

## What we did right

每天开站会，每个人汇报昨天做的以及今天要做的，能及时发现问题和控制风险

后期每天做版本测试，能及时发现问题。如果某个问题当时没发现的话，后面往往要花费好几倍的时候来解决。

## What we did wrong

讨论不够，计划定的不够详细和周全

后期每天做版本测试，但测试比较随性不规范，以后可以列出一个测试项目表（讨论：稳定性crash，性能，显示bug等）

项目内部的交流不够？

编辑器维护得不太好，有很多bug

后期PC版维护不够

每次做版本没有上传版本号，后面就拿不到版本了

后来provison过期导致早期版本机编出来的版本都不能用

对于任务都是被动接收，而且缺少反馈

加班时间比较长，没有效率

有时候与美术的沟通比较困难

## Improvement/Suggestion

每天开站会，形式是否太过单一，是否有更好的方式或者改进

后期每天都要做版本测试 ，有些问题当时没发现的话后面可能要花好几倍的时间来解决

技术分享：是否需要项目期间的不定期培训讨论，每个人介绍下自己做的东西

每个版本都要在p4v上打Tag，上传版本号

以后拿版本都要用版本机的，不要让某个人来编版本。

每个人以后可以负责1-2个模块

对于公司规定的要3-4天WL的情况下才能OT的情况，应该反过来才合理，就是如果周末需要OT的话，本周的WL不能超过3-4天。如果规定不能改的话可以这样，比如这个周末需要OT的话，这周可以WL2天，另外少的可以到下次不OT的那周再补。

另外加班有时候到9点事情就做完，还要等到9点半才能走。

对于每天的站会，细节谈论可以在会后再说。

# Optimizing

优化工作何时进行

性能分析

优化方法

## What we did right

做了比较多的优化工作，优化成果比较好

优化时做了性能分析

## What we did wrong

## Improvement/Suggestion

合理运用工具：Instruments

性能分析比较重要，优化时比较有针对性和方向

# Testing

如何能更方便快捷地重现bug

自动化测试的必要性

## What we did right

QA在TTP上报bug时尽量附上存档，特别是那些比较难重现和比较费时间重现的bug

后期有一个QA来重现bug

## What we did wrong

## Improvement/Suggestion

项目后期需要有一个QA来做bug重现

有些调试做PC版上做比较方便，可以实时修改编译

# Communication

是否出现意见不一致的情况，如何处理

遇到问题时是否能快速寻求到帮助

知道别人在干什么吗

交流成本控制：面谈，会议，文档，etc

任务交接

## What we did right

## What we did wrong

任务交接做得不是很充分

## Improvement/Suggestion

别人做的事情不是很熟悉

# Others

代码规范

技术积累

## What we did right

## What we did wrong

代码规范性不是很好，很多变量函数可以通用的，别人写的代码重用率较低，加的地方也不统一随处都有，UnGame.cpp很大（line num: 7800+）

## Improvement/Suggestion

写代码要做好安全检测和保护，类似数组越界导致内存写坏引起的crash查起来很麻烦

代码要写注释，与原来版本易于区分