# 录像系统设计文档

# 1. 开发目的

大航海项目的测试中经常会出现一些偶现的BUG , 单凭LOG信息比较难定位到问题所在, 于是希望有一种可以完全重现客户端运行时环境的功能, 方便开发人员快速定位和解决BUG, 提高开发效率.

# 2. 设计思路

游戏客户端是一个庞大而复杂的系统, 对内而言, 需要处理复杂的游戏业务逻辑和各种渲染效果, 对外它又要接受玩家的按键的输入还有跟服务器的数据交互. 如下图所示:

鼠标按键输入

数据包交互

玩家

游戏服务器

游戏客户端

(渲染+逻辑)

将游戏客户端形象的比喻为一个游戏世界的话, 那么驱动我们的游戏世界一直持续下去的就是玩家的输入与服务器数据包交互, 试想一下: 在玩家登陆游戏后, 假如我们把玩家的所有输入信息以及与服务器的收发包信息全部记录下来, 然后在下一次启动时, 本地模拟出这些数据按次序丢给客户端处理, 那么客户端所展示的世界会不会跟我们上一次的一模一样? 答案是肯定的. 录像系统设计思路就来自于此.

对于录像系统来说, 存在两种模式: 录制与回放,

1 录制模式下负责记录客户端对外交互的操作, 按照时间顺序依次存入自定义的录像文件.

2 回放模式下, 从录像文件中读取数据, 然后模拟出这些操作丢给客户端处理, 让客户端跑在我们模拟出来的环境中.

录制模式示意图:

游戏服务器

封包数据

封包记录

录像文件

按照时间先后依次存入

游戏客户端

(渲染+逻辑)

按键记录

按键数据

玩家

录制模式下保存用户的按键, 收包发包信息, 存入录像文件中(存入发包信息是为了后面回放模式下做同步处理, 后面会讲到)

回放模式示意图:

模拟封包

封包模拟层

录像文件

从录像文件读取数据

游戏客户端

(渲染+逻辑)

模拟按键

按键模拟层

在回放模式下, 一方面需要拦截按键, 收发包操作, 另一方从录像文件中读取按键, 收发包操作, 并模拟出来丢给客户端执行.

这时, 客户端实际上就像是单机模式下运行了, 发出去的包是没有响应的, 所以模拟发包是没有意义的, 但是在实际的代码实现中, 为了同步录像文件播放进度(使回放录像的结果与运行时的时间一致), 对回放模式做了以下处理: 屏蔽客户端发包, 并记录下当前客户端发包指令, 待执行到录像文件中的发包操作后, 从录像文件中读取发送的数据包数据, 判断与之前记录的发包指令一致, 一致则模拟发包流程. 这样就能让录像回放遇到发包操作时, 实际上是在等待我们的客户端执行到真正的发包操作后, 录像才继续往下播.

# 3. 实现细节

1 ) 按键模拟

修改引擎底层, 监听cocos2dx底层的触摸事件(CCTouchDispatcher::touches), 记录下底层的触控信息.

2 ) 封包模拟

发包跟收包的处理是相同的, 都是直接保存二进制的字节流数据.

3 ) 录像文件定义

将模拟的数据写入文件, 待录像回放时从文件数据恢复出模拟数据返回给客户端执行. 录像文件使用lua语言, 这样做是为省去自己实现词法分析来解析录像文件.

# 4. 遇到的一些技术难点

1) 录像系统对多点触控的支持.

多点触控是手机特有的操作, 这里需要注意对cocos2dx底层多个点击的数据进行记录即可.

2) 录像文件中封包数据的模拟.

在模拟封包数据的时候需要特别注意, 在进行模拟层模拟封包时, 是不关心具体的协议内容的, 只需要根据二进制流恢复封包数据. 实际操作中需要十分小心, 很容易在解析数据包时崩溃.

3) 录像系统对断线重练的支持

航海项目中的断线重练情况比较复杂, 但是处理情况都是大同小异, 分别对断线/重练事件做单独的记录模拟就可以了.

4) 安卓系统下的录像系统

安卓下的录像系统跟PC端是一样的, 只不过安卓下的文件系统相对PC端要复杂一些, 特备是内部存储的读写权限这块, 用户级权限是无法看到内部存储里的数据的,需要ROOT后才可以查看.

# 5 总结

市面上很多游戏会有录像系统的应用实例, 比如大家熟悉的dota, lol都可以保存的经典战役供玩家反复回味, 甚至还有即时观战功能, 实现原理基本都类似.