书承上文 PCR-Timeline-Analysis ver1.2。

各位大佬大家好呀，这里是 **zzpong**。在大家已经了解了各文件数据所代表的含义之后，下面就可以进行游戏的战斗逻辑分析了。这里我依然要先给出这个优秀的链接，详细讲解了角色的战斗逻辑。除了收藏图标是 b 站不好找以外，没啥缺点。但本文会更进一步，告诉你这个完整战斗逻辑的具体细节及其数据调用。

请注意，该逻辑是完全基于数据分析推测得来，并非源码解包，可能与真实情况存在差异。但经过多视频逐帧验证（视频见附件）、初音笔记与静流笔记分析、群中所反馈的基于此模型对“角色极端反应情况”结果的对照，楼主有 90%的信心这就是程序执行流程（除了及其微小的细节外）。有人可能会问了，“那楼主你为什么能这么确定呢”，因为凭借拆若干个 NS 游戏数据、分析 bin 文件十六进制规律推断道具格式、制作 dlc_unxxxxer 的经验来说，还没有失败过（因为金手指和存档修改就是这么做的）。只不过这次从十六进制数据总结道具规律上升为从数据规律总结函数调用方法了。

声明：该分析基于日服数据库与台服游戏端（因为楼主没有日服账号），经测试数值完全一致，但还请不要无端联想。此处仅针对外服客户端内容交流学习，如有雷同，纯属巧合。

在这里首先要感谢**世逝时失**所绘制的逻辑图，以及**群内各位大佬**的讨论分析，没有大家，这一切很难实现。接下来请打开图片 Wait + Action Loop Assumption.jpg，这就是我们说的完整战斗逻辑了：

一个战斗部分包含两种情况：“移动”与“最小行动单元”。其中移动分为“进场移动”及“战斗时移动”，而“最小行动单元”包括“等待”与“动作”。“最小行动单元”是按照固定排序执行的，这个排序可以从“unit_attack_pattern.sql”中获取。**在“最小行动单元”穿插因攻击距离所需的“移动”，就是该游戏的整个战斗逻辑。**

我们可以把“普通攻击”与“技能”都看做一个“动作”[1]。其中普通攻击仅有一个 action，仅会在 unit_id.json 文件中体现；技能则根据描述（比如伤害+debuff）拥有多个 action（同样有描述），这些 action 互相独立且在 skill 释放时同时进行，但触发时间可能各不相同。这些 action 可以在 skill_action.sql 中获取，对应生效时间点可在 unit_id.json 中获得。每个 action 根据自身不同，还会分 visible=0 or 1 的子属性（分别表示“不显示”与“显示”）。一般情况下，仅 action 所描述的属性会被玩家看到，其他属性都只在后台执行（且罕见），所以一般只需要关注 visible=1 且尾部有 action_id 的这些 ExecTimeForPrefab.data.time 即可。

[1] UB 除外，**在自动战斗情况下，UB 会在不打断动作的情况下释放（即与普通“最小行动单元”一同排队释放）**。而**手动时则会直接打断当前“最小行动单元”，并在结束 UB 后立刻进入下一“最小行动单元”。**

上面大家已经大概了解了游戏机理，在详细解释之前，大家还需要了解另一个很重要的名词——“帧”。

游戏画面是通过显卡渲染产生的，而显卡的处理效率并不能生成大家日常生活中感官所见的连续效果（至少目前科技不行，而在技术落后的更久远时还有“隔行扫描”的概念），只能每隔一段时间产生一张图片，这张图片就被称作“帧（frame）”，而“每秒能产生多少张图片”这个值，就叫做“帧率”。大家可能知道，如果在书的页脚画足够多的小人，快速翻页的时候会感觉小人是连续的动态，而非一张张的图片。这是因为你的视网膜像一张可重复利用的老旧相片，每次曝光以后，将有一段时间的暂留效果。在这段时间里，即使没有任何图片，视

网膜依然保留着前一张图片的信息，而这个信息的保留时间是 1/24 秒，即大家常常听到的“24 帧”。当然你会问，为啥眼睛不能像现代数码相机一样咔咔咔连拍呢？这是因为数码相机拍摄完成就可以结束工作，而人眼获取信息后，需要传递给大脑等待处理，所以大脑也并不是一个无限效率的机器，脑机什么的还要等久远的未来。（而有人说自己能区别 30 帧与 60 帧——这只能说是一种感觉，没有对比就没有伤害。但想到绝对音感的存在，能分辨帧率区别的人也是可能存在的，就像能听出 10000 块与 100000 块耳机区别的人一样。）如果你有兴趣拆包视频的话，会发现任何视频都是一连串的图片组成的，游戏所显示的视频也不例外，是由一帧帧的图片所组成的。当然，你可能又会问了，如果录制游戏速率与游戏通过显卡本身产生的帧率不同步怎么办？这时你就需要“帧同步”或者“帧插值”了，在动漫行业，还有一个专门负责绘制中间帧的职业，叫“中割”。以上只是让大家更好了解帧的概念，对于本文来说，只需要明白一点，**录制帧率并不等同于游戏帧率**。所以当你录制视频查询帧数并映射成时间的时候，请务必先了解自己的视频帧率。如果不清楚，请在视频上右键——详细信息。

下面开始详细解释游戏的战斗逻辑：

----------角色移动----------

角色移动是一个非常复杂的环节，其复杂原因在于可能性过多，很难精确命中原始程序，而并非因为逻辑复杂。在这方面的内容上，已经有很多优秀的帖子（希儿天下第一可爱）珠玉在前，也有十分珍贵的视频资料（东方妖灵梦），请点击链接查看。

在接触具体内容前，请大家先了解另一个概念：“状态机”，请联想“数字逻辑电路”或者“马尔科夫链”或者其他什么类似的东西。如果都不知道的话，那就看下面的粗略解释吧：当前的游戏，如果使用顺序执行的逻辑结构，不仅会因为判断逻辑把脑袋烧爆，调试起来还麻烦，效率也低，复用性就更不用说了。所以 unity 采用了另一种并行手段：

每个角色有一个状态，以及各种计数器，一旦某些条件触发，角色状态就会更新（比如“移动——攻击”）。这种方法可以并行处理（宏观上），且符合面向对象逻辑，所以看起来很爽。之所以我这么肯定，是因为我看到普通战斗 dll 文件有这个内容（仅抹方法不抹变量是好文明啊）：

```
// Token: 0x0401032D RID: 66349
[FieldOffset(Offset = "0x8")]
private int <>1__state;

// Token: 0x0401032E RID: 66350
[FieldOffset(Offset = "0xC")]
private object <>2__current;

// Token: 0x0401032F RID: 66351
[FieldOffset(Offset = "0x10")]
public BattleManager <>4__this;

// Token: 0x04010330 RID: 66352
[FieldOffset(Offset = "0x14")]
public UnitCtrl _unitCtrl;

// Token: 0x04010331 RID: 66353
[FieldOffset(Offset = "0x18")]
private float <timer>5__2;
```

看到那个大大的 int state，object current 和 float timer 了吗（程序真的是用秒而非帧为单位运算的）？可能有人会问，楼主楼主，你不是说全靠数据分析来推断问题么，怎么这时候搞这个？你要知道，看到的东西只有方法名没有函数，不就不叫那啥了么，是吧？并且我是真没有源码，要不能花这么多时间猜么 -。-

下面将给出一些必要结论：

1. 本游戏的 y 轴极其简单，五条水平线分别代表五个角色行进轨迹，而这五个角色在组队时直接按照 search_area_width（unit_data.sql）进行排序；
2. 角色距离不计算 y 轴（但某些 boss 是有 y 轴参数的），角色入场间隔 200；
3. 本游戏的角色入场并非从边缘进入，且关卡/地下城/竞技场/公会战各不相同，地图也并非中轴对称，而是己方面积小、出场靠近屏幕左端，敌方面积大、出场据屏幕右端较远，但 jjc 双方移动距离是一致的（目前还没有香港记者出现），因此最右侧一般都会空出来一截。**这些内容**因为**是固定值**，就像“碰撞半径”125 一样，不会写在数据里，而是直接写在程序代码里。如果想要获取，要么用尺子量（不同手机可能不同，请计算比例），要么挖代码；
4. 角色索敌范围是 search_area_width（unit_data.sql），角色移动速度为 move_speed（unit_data.sql）（1 秒移动 move_speed 个单位，**不是 1 帧**）；
5. 角色攻击距离 = 角色索敌范围 + 碰撞半径，角色间互相独立，一直集火**敌方最前可见角色**（仅计算 x 轴，没有画个圆看哪个近的概念）。

有了上面数据，基本上就可以进行计算了。

还有一些 FAQ（纯属娱乐，如有雷同，我也不知道怎么说了）：

1. 为什么要有碰撞半径，碰撞半径为什么是 125？

答：可能测试版本是没有 125 的碰撞半径的，直到**羊驼**的出现。如果你写了 100 行的 unit_data.sql，老板跟你说“我们觉得最近羊驼有点火，还能照顾 furrrry 党，加个肉弹冲撞到游戏里试试。既然战斗开始后登场，那肯定就得后来居上承受伤害啊，你就把它索敌范围调的小小的。不行，你没有 get 我的 point，我要最小的，就比那个第二小的宫子还要小。”“好的老板”，你看了看**宫子的攻击距离 125**，那就 105 吧。忙完一切你在喝咖啡等待下班，老板拍了拍你肩膀，“你这不行啊，年轻人努力是福报，你这改了参数都不注意的？双方的羊驼都要贴在一起了！你还喝咖啡？工资扣一百，明早我要看新结果，不明早了，今晚十点前给我发过来（虽然我不会看）！”然后你对着 100 行的 sql，一个个调参数？又累，改错了还要被老板骂，那咋办？其他角色参数都好好的，那我**把角色整体平移下**不就好了？sql 太难改了，就代码那攻击距离公式后面加个常数吧。要能容得下一只羊驼，那总不能加羊驼自己，那就加第二的宫子吧。**所以碰撞半径是 125**。

2. 我看过视频了，为什么攻击方优先放 UB，为什么普攻有时同步，有时防御方优先，并且从没有攻击方优先呢？

答：你已经了解到上面“状态机”的概念了（当然 unity 业内人士可能不这么叫，还请见谅）。jjc 一共有十个角色，也就是有 10 个独立的个体，分别计算执行，互不干涉，其“计步器”也都在独立作用，等着“小样进我攻击范围了吧”的那一天。所以当完全一样的镜像角色互相打的时候，不仅普攻，动作也是完全同步的。

这时你的老板又出现了“你这个不行呀，我找了两队镜像角色互殴，那谁先放 ub 呢？你不要羡慕我独到的目光，我是你的五彩斑斓黑。”本来你想掀桌子就走，但想到房贷车贷，你忍住了：“那老板你看这样如何：**既然 jjc 战斗只有进攻方玩家能看见，那我就先让进攻玩家爽 ub**，玩家爽了氪金了，老板你不就爽了？”“不错啊小伙子，100 块不给你扣了。”老板高兴地坐着宝马离开了。

虚惊一场的你现在有些小得意，毕竟转危为安，还赚了 100 块。但随后你发现了一个问题……

程序时间是按秒算的，spine 骨架播放也是，那要是有些角色走到索敌范围，刚好迈出腿怎么办？总不能金鸡独立下一帧直接站定攻击吧，虽然角色设计师看不到索敌范围，没法考虑修复，但我不行啊。那角色互退一步，岂不是打不到；互进一步，都这么近了，下一帧还是迈出腿怎么办，再等的话，角色是不是跟羊驼一样亲在一起了？哎呀，这么想想，100 块好像要飞了啊。等等，那我**让一方先停止不就行了**？哪一方呢，进攻方已经在 ub 上吃了优势，要是无人可守，玩家不开心，老板不开心，我就被迫不开心了啊。得，**那就让防御方先站定，攻击方牺牲下，找个正常位置停止吧**，所以防御方**在这种情况下**会先攻击。这一折腾可不得了，站队需要排头兵，攻击方排头兵受了委屈乱了，后面队友也就一并乱了。除非……JOJO，我不当排头兵啦（指黄骑）！

----------角色动作----------

相比于角色移动的盲人摸象，角色动作可谓逻辑上的清流。

首先，你访问 **unit_data.sql** 获取角色 id，normal_atk_cast_time（普通攻击等待时间）、search_area_width（好像这时候已经不需要了）；

其次，你访问 **skill_data.sql** 获取角色技能等待时间 skill_cast_time、角色技能范围 skill_area_width、skill 的 action 列表 action_1、action_2、blabla；

然后，你访问 **unit_attack_pattern.sql** 获取角色的行为逻辑（不知道怎么组合的请复习上文）；

最后，你访问 **prefabs.zip（干炸里脊站下载）或 trans.json（附件）**获得角色 skill 的 action 的对应 ExecTimeForPrefab.data.time。

好了，角色动作已经完成，剩下的就是等待游戏播放行动动画了，是不是很简单呀？

FAQ:

1. 行动动画没有播放时间吗？

答：目前没找到，并且我更倾向于 spine 动画结束后返回一个值，直接触发并修改状态机的状态（动作——等待）；

2. 技能或攻击生效时间看的是哪儿呢？

答：看 ExecTimeForPrefab.data.time 前有没有 ub 打断，有的话就全部木哒哒；

3. 有些角色的都市传说怎么解释呢？

答：请阅读上文最后部分。

以上所有内容均仅依据游戏本身以及所获得的数据文件推测而来，并未涉及或阅读任何代码部分，请酌情参考。最后祝大家游戏愉快~

详细文件分析请移步另一篇文章，PCR-Timeline-Analysis ver1.2。
