在经过了两天对数据包的挖掘，进度已经到了 PCR-Timeline-Analysis-Package ver1.2。虽然文件数据的拆包分析已经被游戏函数对象部分验证过了，但时间轴的整体逻辑依旧未经实例测试，虽然据群友反馈“相差不大”，但本着精确无 bug 的原则，我还是自己录制了几个视频进行测试。其中视频录制选用的 mumu 模拟器自带的“录制视频”app，游戏为台服账号。录制后视频通过“文件共享”导出，并通过 Adobe Premiere Pro CC 2017 打开。

在刚开始的时候，我以为 PR 的时间轴显示的数值就是时间本身，所以是这么写的：

```
3:13  站定，喊口号，立即触发。
4:01  插地板，buff成功贴上
4:11  口号消失，buff保持，buff时间与动作无关，而是参数。
5:04-5:13  站定  3.13+1.79（喊口号，技能动画开始）+0.1（与口号消失时间不一致，消失为0.98，不过这个不重要，可能只是一个口号罢了）  = 5.02
7:19普攻
8:21站定

31-13=18帧 口号-buff   即0.3*60
动画120帧    1.79+0.1=1.89*60=113.4

2*60+6=126
60+2=62

cast time 0
datatime  0.6  0.6  0.6  表示同时触发。
cutinfadestart   1.7999999998
cutindura      0.1
cutinskip   0
```

当时我觉得这时间是没问题的，你看 3.13+1.79+0.1=5.02，掐头去尾不考虑误差，可以说恰到好处。这时又恰逢群友讨论最热烈的“cutinxxxx 为动画时间”，而这个计算又“证明了这一点”，感觉一切谜题已经解开了。但随后我又录制了一个视频进行测试，却发现：

52.02 莫妮卡她动了！
52.12 伤害值+吸血
52.27 伤害值+吸血展示完毕（下一次展示了：30帧，所以这个是随机的）
53:07 后跳+收刀
55:19 莫妮卡她动了！
55:29 产生伤害+吸血
56:24 后跳+收刀

普攻间隔 2*60+12=132 实际2.24*60=134.4
产生伤害值10 实际datatime 0.3*60=18
动画时间
1*60+5=65 实际1.9+0.1=2.0*60=120

24-19*2=10+60=70

你录制的是30帧，而实际是60帧。

新视频不遵循这个规律，或者说唯一准确无误的“wiki”信息所提供的“行动 buff 仅会影响待机时间，不会影响动作时间”，也变得不准起来，甚至“有 buff 的待机时间”要更长一些。如果说“第一次 buff 在这个动作的时间点并没有生效（日社员工的慵懒，buff 提供的属性不是动态的，而是在“进入动作前”仅读取一次当前属性”，所以请务必先用 nnk ub 再用春田 ub），但之后的多次测量结果也不符合这个理论。

1:02:19
1:03:29　60+10=70帧

1:06:12
1:07:15　60+3=63帧

4:15-4:21　6帧，慢慢抬起刀，动作0.1s的渲染，0.1*6
4：21 - 7:04　2*60+47=167帧等待
8:12　68帧动作

11:25待机
13:12挥刀 107帧等待
14:19　67帧动作
16:02　103帧等待

17:09
18:01　锁定射击

3:13

6:28 静止　2.83
9:11 攻击　2:43　134.4
10:19静止　1:08
11:07锁定射击　0：48——0.86*60=51.6
13:12静止　2:05
14:26攻击　1:14
16:04静止　1:38
17:22攻击　1:18　78　2.24 *60=134.4
18:27静止　1:05

19:15　22:00

4:15　5:04　50帧
6:22　127帧

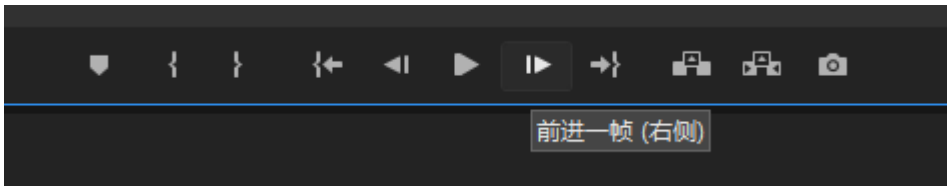
9.07　10:23　1:16　1.55*60=93
11:06　1.55+0.8=2.3*60=130x

于是我又录制了更多的视频，测试了更多的角色，并尽量选择“没有行动 buff”的裸角色对抗“没有减速效果”的敌方角色（所以才选择了伤害 ue）。但问题还是没能解决，除了

cast_time 外，所有的一切（action.data.time，cutinxxxxtime）都对不上，并且 cast_time 也显得奇奇怪怪的。

于是就像上面两张图所展示的一样，我发现 PR 所提供数值的最后一位总是小于 30。难道这不是时间而是帧率吗？我又去翻别人做的视频，然后思考，是不是“自己录制为 30 帧，实际上为 60 帧”呢？视频里又为什么有时候会说 PR 这个数据叫“帧”呢？如果是 60 帧，那 PR 数据的 xx：yy：zz 是不是应该计算 xx*3600+yy*60+zz 呢？那我的 zz 上限只有 30，zz_current - zz_previous 又需不需要乘以 2 呢？基于这些的推断，我就像上图一样反复测试计算数据，但总是不对。

后来我就去搜 PR 的数字到底是什么含义，并且我明显看到该数字是与当前播放进度一样的，应该就是时间了。那能不能切换成“帧数”方便计算呢，查询无果。直到我点在视频播放的箭头上：



如果我丢弃之前关于 PR 数据的所有不可靠猜想，仅依靠这个箭头，是不是能解决一切问题呢？是可以的。但如果仅有这个功能，那我们需要找的就是“相对帧数”，而不是“绝对帧数”了，并且这个“相对帧数”需要尽量小，来减少肉眼判断的误差影响（现在想想太小也有问题，因为会被 floor 掉）。

于是我看到了莫妮卡的二技能“锁定射击”，该技能是分两部分，“伤害”与“眩晕”。而这些在之前都已经被分析完毕，visiable type 表示伤害可见，于是就有了下图：

visiable type 1.55和1.6，分别是伤害和降防。
10:22-10:23单位为帧，因为箭头写着“前进一帧”。 1帧 - 0.05 20帧-1
1.55*20 = 31.0帧，向前找31帧，是不是就刚好喊口号了呢。。。。（分毫不差，31帧前刚好喊口号）（没buff是33帧）
那cutin 1.9+0.1=2，是不是就是40帧动作呢？不对，是47帧。
那攻击cutin是不1.9+0.1=40帧呢？也不对，是24帧，1s。说明这个参数真的没用。（并且连被动都有这个参数啊 2333）
那攻击前摇是50/52帧么？2.24*20=44.8帧，没错了（大概）。
那攻击产生伤害是0.3*20=6帧吗，不是，是9帧

有buff等待48，没有等待49。如果把cutinfade拿到这儿，0.1*20=2帧。剩下的是动作38帧吗？不对。

如果不是精确的20帧，而是24帧呢？

锁定射击是50帧。24帧*2=48。。。。

24帧，才会有这么多小数啊，还是循环的，因为有3。

空花已验证

可能也是运气好的原因，我所录制的视频刚好有这个黄色的“伤害”数字，以及 70 级莫妮卡打 150 级 ue 等级差导致的“眩晕”miss 文字。而它们之间，仅间隔 1 帧，就是点一下“前进一帧”就从跳“伤害”变成了跳“miss”。于是我写下，“1 帧=0.05，那么 20 帧=1”。

在检测多个角色，重新查看 cutin 时间的时候，我发现每个技能的 cutin 都差不多，然而手动按“下一帧”数帧数会发现，不同技能的时间是完全不同，甚至有些技能间都是两倍的关系。也就是说，cutin 并不是技能动画时间。

在确认并果断舍弃 cutin 数据以后，所有的数据几乎都对上了，误差在 3-5 帧之间，这时候一般都会直接将这个看作“肉眼对角色行动判断的误差”。

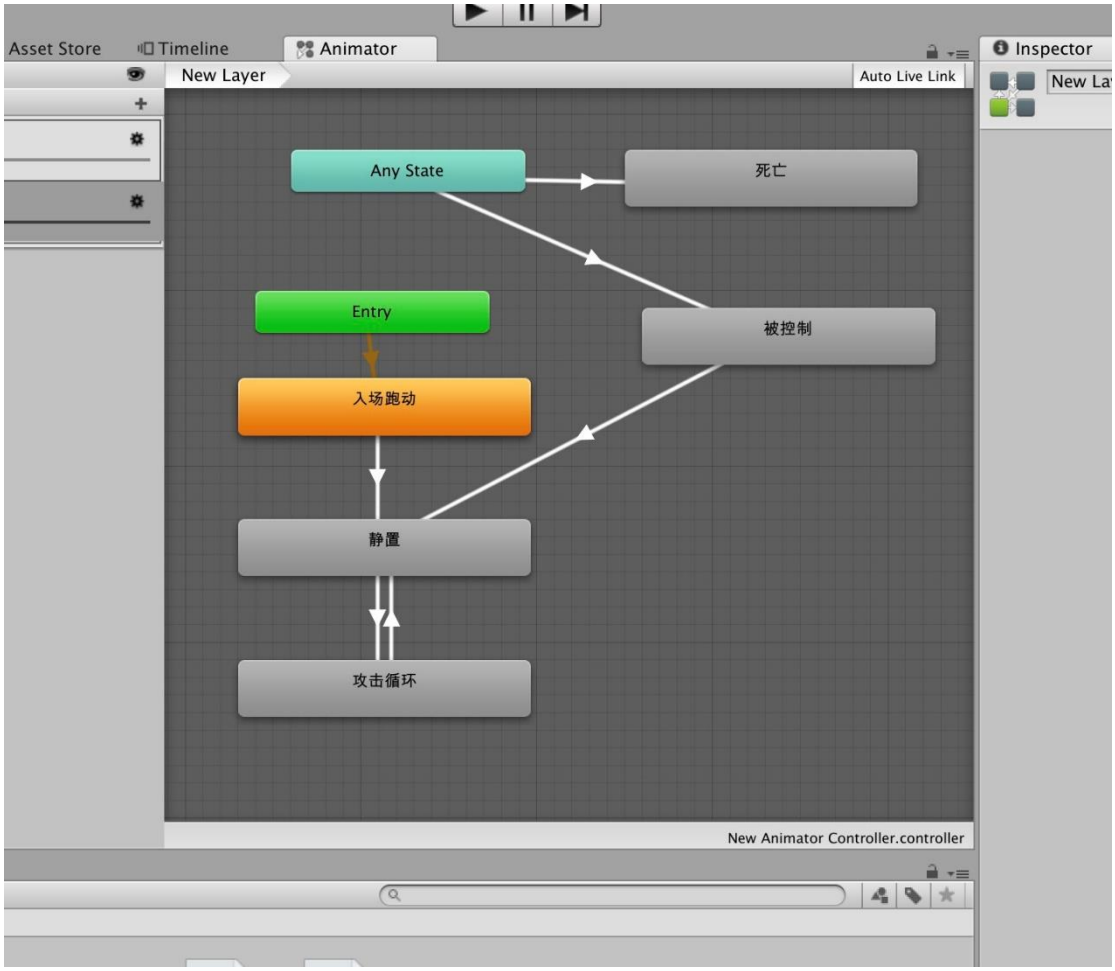
但我还是觉得奇怪：如果 20 帧=1 的话，为什么游戏中的各种“time”会有循环小数呢？2 是一个可以被整除的数字，而我常见的帧率也只有 29.97、30、60 这些啊。于是我便去网上搜索“不同设备录制的视频帧率不同”，随即我立马看到了这个：“动画、电影等视频帧速率有每秒几帧到几十上百，甚至每秒几千帧（高速拍摄）。只是 24 帧/s 的普遍些。”没错，**24 帧与 20 帧，floor(0.1*24) = 2 = 20*0.1**。矫正这个问题后，所有的帧与计算值都更加接近，甚至有些已经分毫不差的对齐了，包括之前的“那攻击前摇是 50/52 帧么？2.24*20=44.8 帧”，2.24*20=53.76 帧。这也让我明白，因为 24 的公约数 3 的存在，使得游戏数据文件中记录了好多 3.66666667 之类的值的原因。

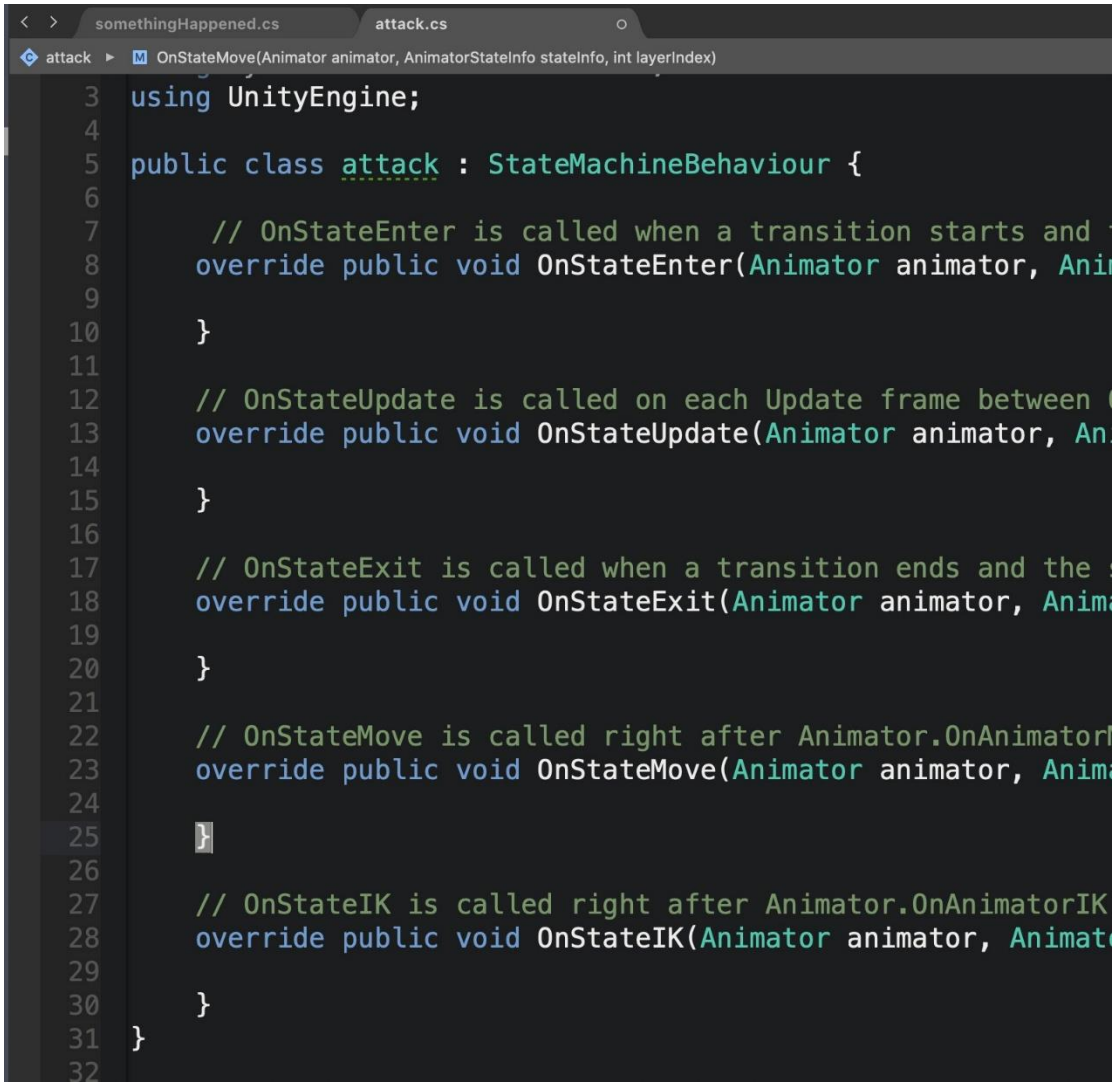
经过上面分析，在绕过帧率以及 cutinxxxxtime 的坑之后，其实大部分的问题都已经明了。但还有一点，如果 **prefabs.zip 的 json 文件里都没有角色动画时间的话，那它在哪**？为此我又重新翻了一遍 sql 文件，一无所获。如果想要角色攻击循环，不需要预定一个角色动作时间的值吗？单凭动画本身程序会不会不稳定呢？在想到这儿的时候，我突然反应过来：如果动画只是动画，播放结束后函数返回一个“结束标志”，然后系统开始调用下一个 action 呢？那 **prefabs.zip 的 json** 所给出的各种伤害出现时间，实际上就是在“伪造一个即时演算的假象（后来反应过来，老游戏好像都是这么做的）”，攻击造成的伤害根本没有所谓“物理弹道”，而只是“在 data.time 时间后在屏幕上显示伤害”，再加上一些“屏幕颤动”、“模糊”之类的特效（没错，这些全都在 **prefabs.zip 的 json** 里）。也就是说，**播放动画与伤害效果是两层完全独立的内容**。这么一想，那 **wiki 上所说“行动 buff 仅影响待机时间”也能解释通了，因为待机是有时间的 cast_time，乘个系数就好了；而行动则全是播片与预制（prefabs）攻击效果，怎么延长，视频可以冒着卡死的风险插帧、预制效果怎么插（日社程序员の慵懒）**？

在一切都想通以后，我突然想到了之前在群里说的“大胆想法”——动画播片是独立的，cast_time 到了就播片。但还不够大胆，因为我当时觉得至少应该做个“播片时间”的双保险，避免程序出现问题。不过之后群里了解 unity 项目的大佬**世逝时失**在我“大胆想法”后说了这么一段话：“其实在 Unity 里的角色控制是酱紫的（下图）。然后每个状态都可以播放特定角色的动画，并且设置转移条件。初期以外，每个 state 可以挂一个脚本。而这个脚本，默认状态长这样（下下图），你可以看到这个脚本有几个 hook，而这个 hook 对应的是每个 state 进入时的逻辑，Unity 开发写代码都是这个思路。”

也就是说，如果我懂 Unity 或者稍微多注意下大佬的话，在**角色动画**问题上就能省下至少两小时的思考时间……不过所幸结果还是好的：时间轴动作全部摸清，剩下的只需要大佬去写代码了。

最后，公会战又开始了，该熬夜了，兄弟们~


Asset Store
Timeline
Animator
Inspector
New Layer
Auto Live Link
Any State
死亡
Entry
被控制
入场跑动
静置
攻击循环
New Animator Controller.controller


somethingHappened.cs
attack.cs
attack ▸ OnStateMove(Animator animator, AnimatorStateInfo stateInfo, int layerIndex)
using UnityEngine;
public class attack : StateMachineBehaviour {
// OnStateEnter is called when a transition starts and
override public void OnStateEnter(Animator animator, Ani
}
// OnStateUpdate is called on each Update frame between
override public void OnStateUpdate(Animator animator, An
}
// OnStateExit is called when a transition ends and the
override public void OnStateExit(Animator animator, Anim
}
// OnStateMove is called right after Animator.OnAnimator
override public void OnStateMove(Animator animator, Anim
}
// OnStateIK is called right after Animator.OnAnimatorIK
override public void OnStateIK(Animator animator, Animat
}
}