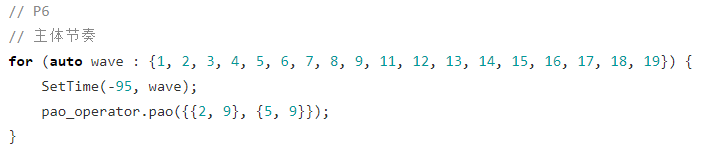
**一、前言**

大概去年10月份的时候，鄙人突然对PvZ脚本起了兴趣。在发现五车场地的无炮挂机无甚进展后，便选择了雪叔@雪漠北 的核聚变无炮，开始用avz尝试写第一个五车场地的无炮挂机脚本。写了个慢速关1f的半成品后，便开始来贴吧发贴，这就有了<https://tieba.baidu.com/p/7598086677> 这个贴子。后来到11月中下旬，脚本已经大体完成，只剩下亿些修修补补的工作。然而，当时现实生活已经越来越忙，而且我对做修补工作实在缺乏兴趣，这个脚本也就放下了。到今年快6月的时候，@刚刚雨纷纷 突然来联系我，说是想看看脚本，这事这才有了下文。后来，经过了他和b站@风若轻雨 的挂机检验（100f挂机测试视频：BV1dT411c7JG），这个半成品脚本的稳定性得到了初步的验证。刚刚雨纷纷又鼓励我说，即使是半成品，里面的内容也足够发一个贴子了（他现在说，能打100f已经算成品了）。于是我就抽空写了这篇内容，来详细讲讲本无炮脚本，或者说绝大多数无炮脚本，应该写什么，又应该怎么写。

**二、为什么是核聚变**

写无炮脚本，与写炮阵脚本的最大区别，就在于节奏。无炮阵的节奏由于其固有特性，存在相当大的波动，这就使得炮阵脚本的常规写法在无炮阵这里很难有所发挥。对于这个问题，其实有两种解决办法。一种是选择一个尽可能精确的无炮阵型，为其编写一套尽可能精确的节奏，从而实现使用类炮阵脚本进行无炮挂机的效果。另一种则是接受并适应无炮阵的波动性，并用全新的框架实现无炮挂机。个人对无炮的理解比较浅，因此第一种思路一上来就被我放弃了，这里采用的是第二种思路。当然，尽管有点怀疑可行性，我还是非常乐于见到用第一种思路制作的无炮挂机脚本的。



（图为炮阵脚本的常规写法）

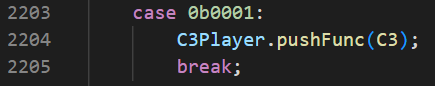
既然是要适应无炮波动性来写脚本，那么选择核聚变无炮作为载体的理由就比较明显了，那就是它的容错性。节奏波动容易导致我在自己浅薄的无炮阵型储备中翻找，似乎也没有找到第二个可以全难度带曾哥套件的五车场地无炮阵型了，而自己也尝试了下这个阵，感觉就连我这种手残玩家也能玩个大概。因此我认为，作为对五车场地无炮挂机的第一次尝试，核聚变无炮是一个非常合适的载体。

在这个部分的最后，谈一件小事——车底炸。延申开来，也就是说，脚本能力的边界该如何界定。个人对脚本的限制是比较严格的，比如说车底炸这事，我在脚本里其实就已经在有意去避免了。这点在我自己的视频（BV1cv411u78w）的p5里的樱桃波上体现的比较明确。那里面樱桃爆炸的时候，前排花盆并不会被冰车碾压。当然，就实际情况来说，这种事并不能每次都做到。灰烬被提前引爆还可接受，但灰烬放下去秒炸其实就已经有点犯规的嫌疑了。总得来说，我认为，脚本能做的事情与人能做到的事情不应有太大的区别，比如说，脚本可以立马得知小丑有没有开盒，但不应该知道小丑还有多久开盒。因为后者对人来说，是无论如何做不到的。但是其实这么界定也是有些许不严谨的，比如，人理论上能精确统计场上的每一只植物和僵尸所受的伤害，那脚本是不是也被允许做这事呢？对此我还是持保留意见的。

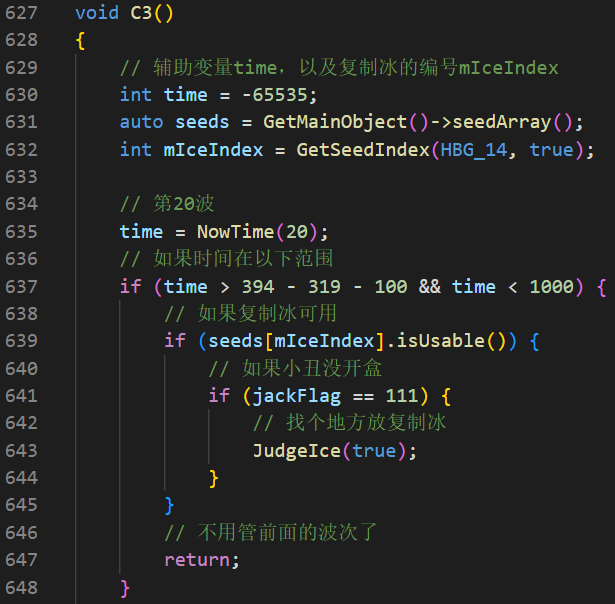
**三、从精确逐波讲起**

这里还是要先讲精确逐波，即之前提到的炮阵脚本的常规写法——脚本作出规定，在哪波的哪个时间点，做哪些事。先讲它是因为它在脚本中有应用，也足够简单，能够让读者热热身，以及对脚本有初步的理解。写到这里发现我还没给脚本链接，那么请从 <https://github.com/Nihilslave/Nuclear-Fusion> 这里下载脚本。

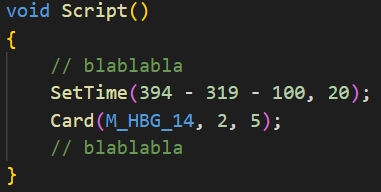
来看最简单的情况之一——纯橄榄无冰车快速关：



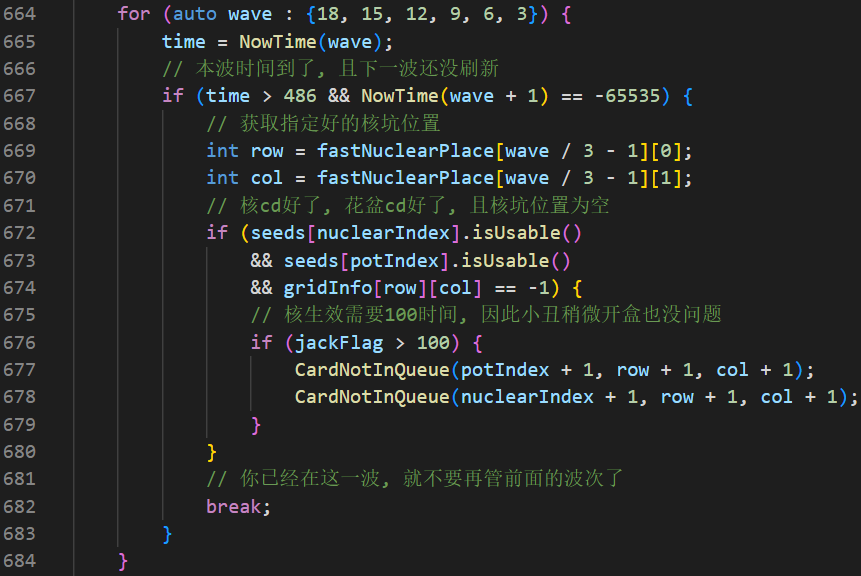
我们在TickRunner C3Player里push了函数C3。这个C3Player是脚本中负责快速关的TickRunner，每帧会运行一次C3函数。特别强调一下这个C3是每帧都会运行的，这对后面理解脚本很重要。相信从这里大家就感觉到了一些不同——虽然都是精确逐波，但炮阵的写法是直接在script函数里把整局游戏的“时刻表”都写好，而我们这里却选择进行逐帧判断。看下去，相信大家能明白这样写的理由。



来到函数C3内部，我们首先会发现这里打的竟然是双冰。不过打法具体是啥其实并不重要，我们主要关注写法。来看对第20波的处理，这里的三次判断是明显不同于炮阵脚本的地方。首先它不要求在精确时间用卡，而是给了一个时间范围。因为无论规定了什么时间，在那个时间点，由于无炮的波动性，总会出现意外用不了卡的情况，而过了那个时间点后，这张卡就不会再被用了，这是不能接受的。其次它对用卡时的场上状态提出了要求：卡要有cd，而且小丑不能开盒（否则有被炸风险）。这同样是因为无炮的波动性，导致很难从节奏上去直接保证小丑不炸，因此需要对此进行额外的判断。最后它没有指定一个放卡的固定位置，而是根据具体情况来判断。这是由屋顶场景的特殊性决定的，并不是每次你想放灰烬的时候，都有格子给你放。花盆cd好了，可以放个花盆；花盆cd没好，就只能把大喷铲了。那么，作为对比，在常规炮阵里，类似的东西可能是这么写的：



我们再来看周期性的节奏，这里以N为例：

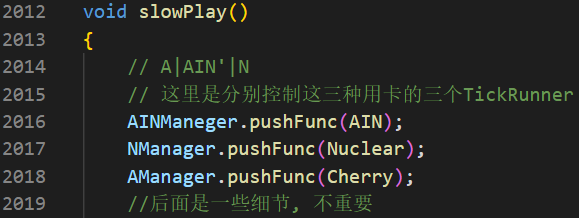


这里的主要讲究在于波次的判断，如果跟20波时类似地写time > xxx && time < xxx，则可能会受提前刷新的影响。比如你写time < 1500，那可能1400的时候，下一波已经来了，这个时候再放核，节奏就可能乱掉。当然，这里更严谨的写法应该是NowTime(wave + 1) < 0。因为对于炮阵脚本，在一波刷新倒计时的时候进行操作，还可以解释成预判炸；而对于无炮脚本，这么解释就多少有点牵强了。

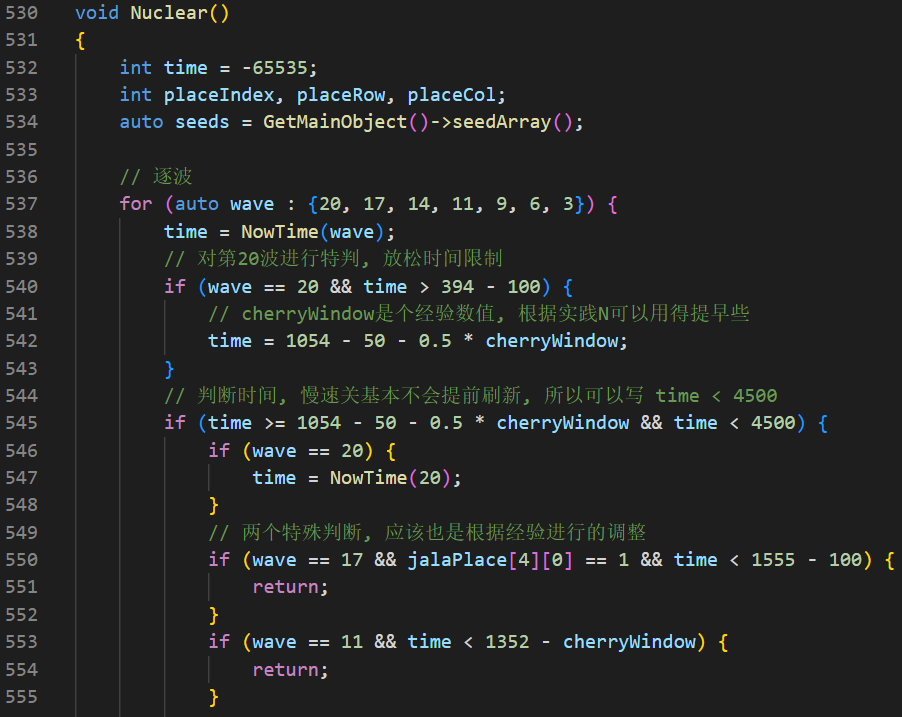
精确逐波的写法，到这里算是讲完了。不过有人可能会问，这脚本明明没有给出精确的用卡时间，用卡的时候还要这样那样的判断，到底“精确”在哪？这个嘛，相信你看了后面的其它构型之后就会理解了。简单来说，叫它“精确”逐波，是因为它的思路跟精确脚本是一脉相承的，只是根据无炮的实际情况做了一些微调而已。而后面的内容，则可以称为“魔改”了。

**四、波动逐波**

使用上面的思路，相信你已经能够写出一些无炮阵型的快速关的挂机脚本。那么接下来我们来看慢速关（出怪同时含红白）。与快速关跟变速关不同，慢速关的出怪波动问题并不显著，这里我们面临的主要问题是僵尸死得太晚。这会导致对前排南瓜的频繁铲种，以及可能的曾哥被砸，从而浪费大量阳光。那么，怎么让僵尸死得及时一点呢？根据我粗浅的无炮理解，解决办法是早点用卡，或者，更正式地说，压缩循环时间。让我们来看具体实现：



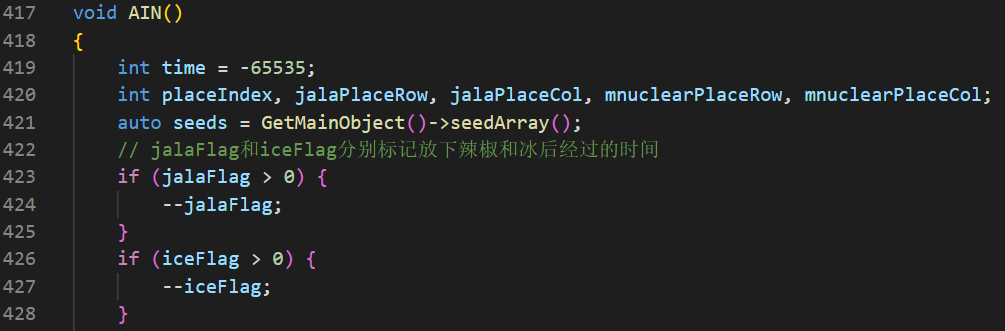
可以看到我们打的是A|AIN’|N的C3节奏，这里可能用核弹开更好些，但这不是我们关注的重点。这里分了3个函数来分别处理这3种用卡，其实只用1个函数也可以，这里主要是想让代码清晰一点。我们先来看这里比较简单的N的实现：





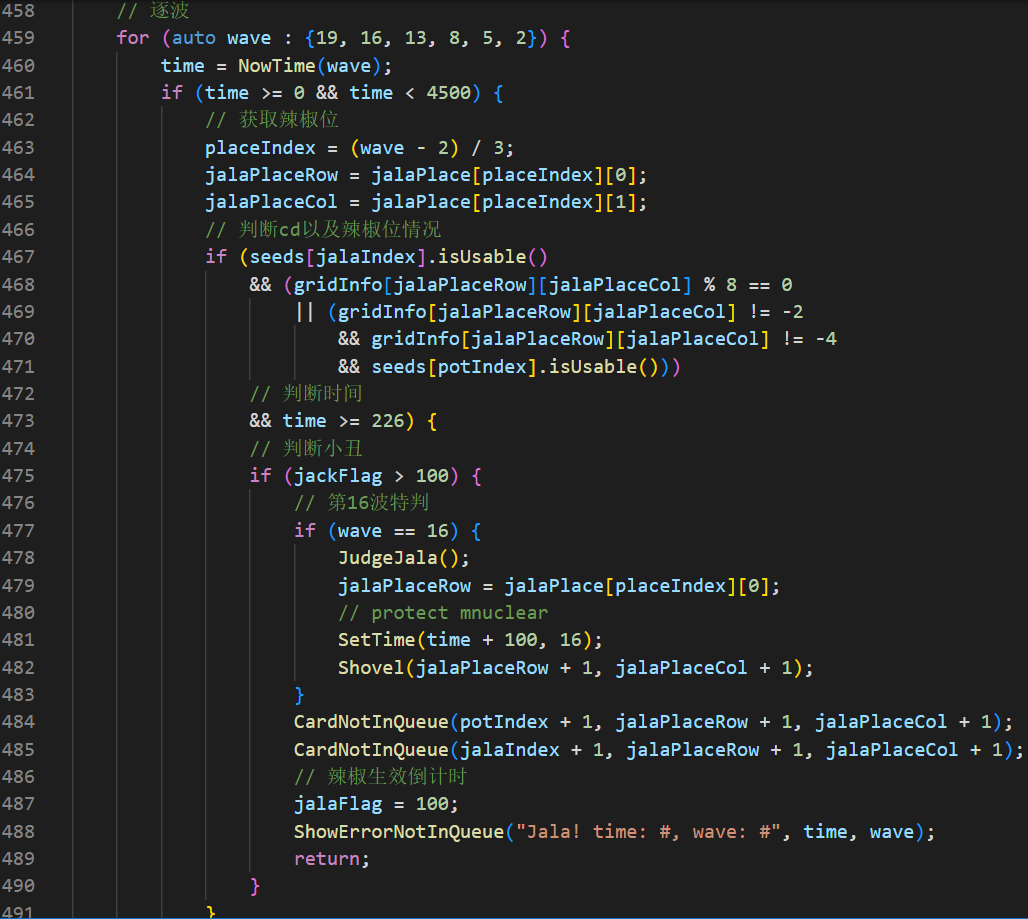
第一张图完全是一些细节，这里不去管他，我们主要看第二张图。我们看到，这里的写法与精确逐波有两点重要的不同：判断下一波卡的cd，以及控制花盆。后者依然是屋顶场景特性，这里真正重要的是对下一波卡的判断。前面说到，我们要尽可能地压缩循环长度，那么对于循环长度的压缩，怎样才算到了极限呢？这里我的解决方法是，判断下一波的卡能不能在合理的时间接上。有人可能会想：慢速关刷新时间稳定，每波都按照规定时间差不多地来，自然就能接上，怎么会需要这样的判断呢？这里有两个理由。一是由于樱桃放在6列而非7列，樱桃炸过后有时不会立马刷新，如果对此不做处理，则会有僵尸死得过晚的问题。而若要对此做处理，则就需要讲之后的用卡时间提前，而如何决定提前量的多少呢？这就需要我们进行这样的判断。二是放灰烬需要避开小丑爆炸，这也会对节奏产生微小的扰动。

正是这个对下一波卡的判断，把原本独立的一波波用卡串了起来，并使它们有了一定的自我调节能力。这也就是“波动”逐波中“波动”之所在。接下来我们再往深处走，看一看AIN’的实现。





先看比较简单的第20波特判。这里主要体现了AIN’这个用卡的特殊性——三张卡是连在一起用的，而且有一个相对固定的时间间隔。而第20波不用放最前面的那个A，场上的情况也相对简单，可以作为热身来看。为了实现这个时间间隔，最基本的，是要在放I的时候判断N’的cd是否快要好了，以及核坑的情况是否理想。一切检查完毕之后，就是把I放下，然后预定一个90时间后的N’。但是，相信大家也看到了，我在图里445行说，“可以直接“预定，那么，什么情况下，我们不可以直接预定呢？



在辣椒的使用上，这里脚本没有直接在100时间（即辣椒生效所需时间）后直接预定一个冰，而是给出一个倒计时。这是因为无法预测100时间后小丑的开盒情况。如果这里定死时间，则复制核有被小丑炸掉的风险。我们继续来看IN’。





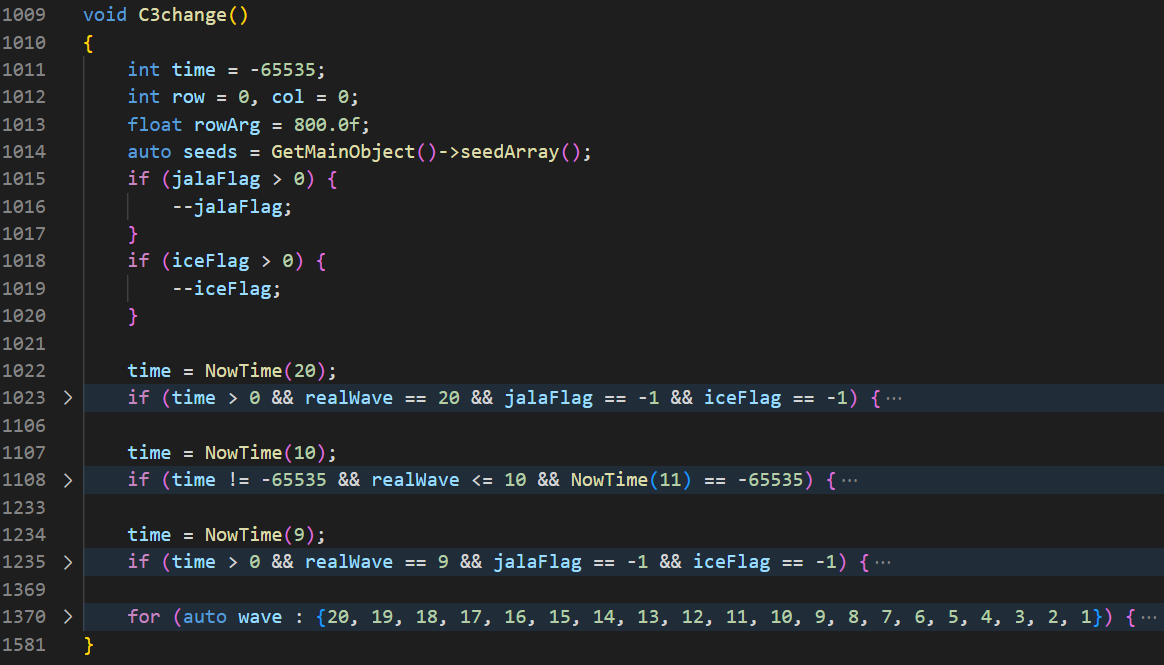
我们发现，AIN’这三张同一波的用卡之间，也通过倒计时的方式纠缠起来了。这里其实有一个问题我一直没有想清楚，那就是I和N’之间的时间间隔到底应该是多少，才能尽量避免复制核被小丑炸。可能以前我是想清楚了的，但总之现在我忘了。脚本里用的90仅供参考，还希望有人能论证下到底什么数值比较好。

至此，波动逐波的例子讲完了。总得来说，波动逐波的思想，就是让不同波的用卡，以及同一波的不同用卡之间产生关联，从而达到动态调控波长等目的。对于慢速关这样不容易提前刷新的出怪类型来说，波动逐波应能有不错的发挥。

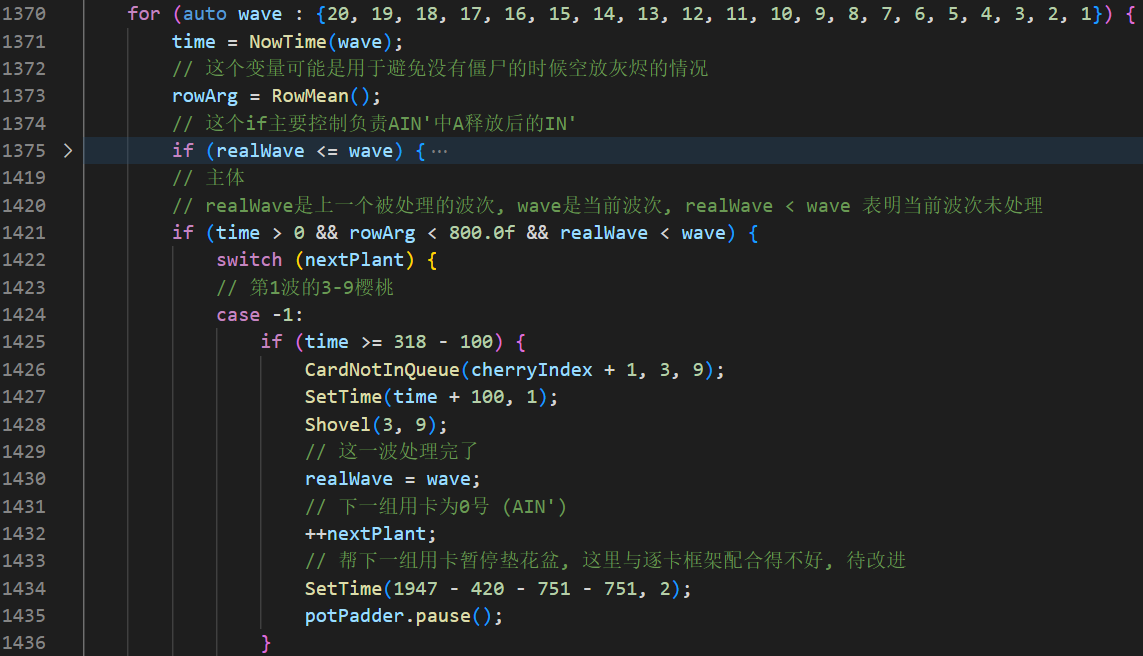
**五、顺序逐卡**

最后我们来处理变速关，或者说，我们来处理提前刷新。我们之前的波动逐波框架，虽然有一定的调节机制来应付延迟刷新，和一定程度上适应提前刷新，但变速关中量大管饱的提前刷新仍然是波动逐波框架扛不住的，这会导致两个致命问题——循环过载和用卡紊乱。循环过载指一个大循环时间显著短于5000，导致灰烬的cd恢复不过来；用卡紊乱指在不合适的时机用卡，比如说把前一波的卡和这一波的卡同时在这一波放出来了。

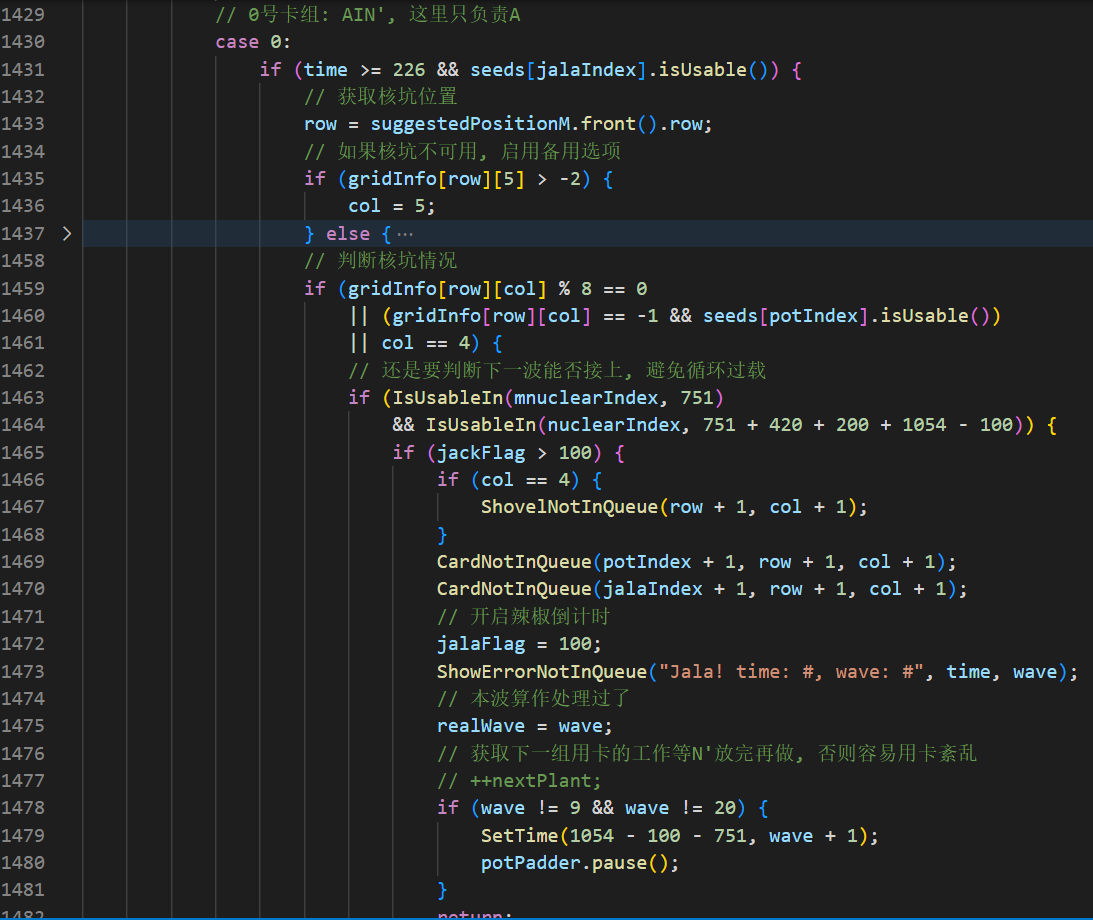
那么，大家对解决这两个问题有什么好的思路呢？我们不妨问自己这样一个哲学问题：脚本到底应该怎样打游戏？这个问题，我认为，可能是大量的脚本写手完全没有想过的，因为大多数人的思路都被常规炮阵脚本限定死了。“脚本怎么打游戏？按时刻表打就好了嘛。”那么对于这个问题，我的回答是，人怎么打，脚本就怎么打。大家在打无炮的时候，碰到提前刷新，都是怎么处理的呢？比如说，第14波本来该用A，结果还没来得及用呢，第15波就刷新了。那么这个时候，一般情况下的处理，就是对第15波用那个原本该在14波用的A。那么16波呢？A接下来不是该用AIN’吗，那就用AIN’处理16波好了。这就相当于把之后的用卡都往后延了一波。在这之中，有一个关键的思想——现在是第几波不重要，现在该用什么卡才重要。也就是说，我不管现在是第几波，也不管有几波我没来得及处理，我上一次用卡用了A，那我这次用卡就得用AIN’，下次用卡就该用N。由此，我们顺序逐卡框架的基本逻辑就是——我现在该用卡吗->该用，那我现在该用什么卡->用好了，下次用下一组卡；下次具体是什么时候呢，我不关心，我只需要不断问自己该不该用卡就行了。



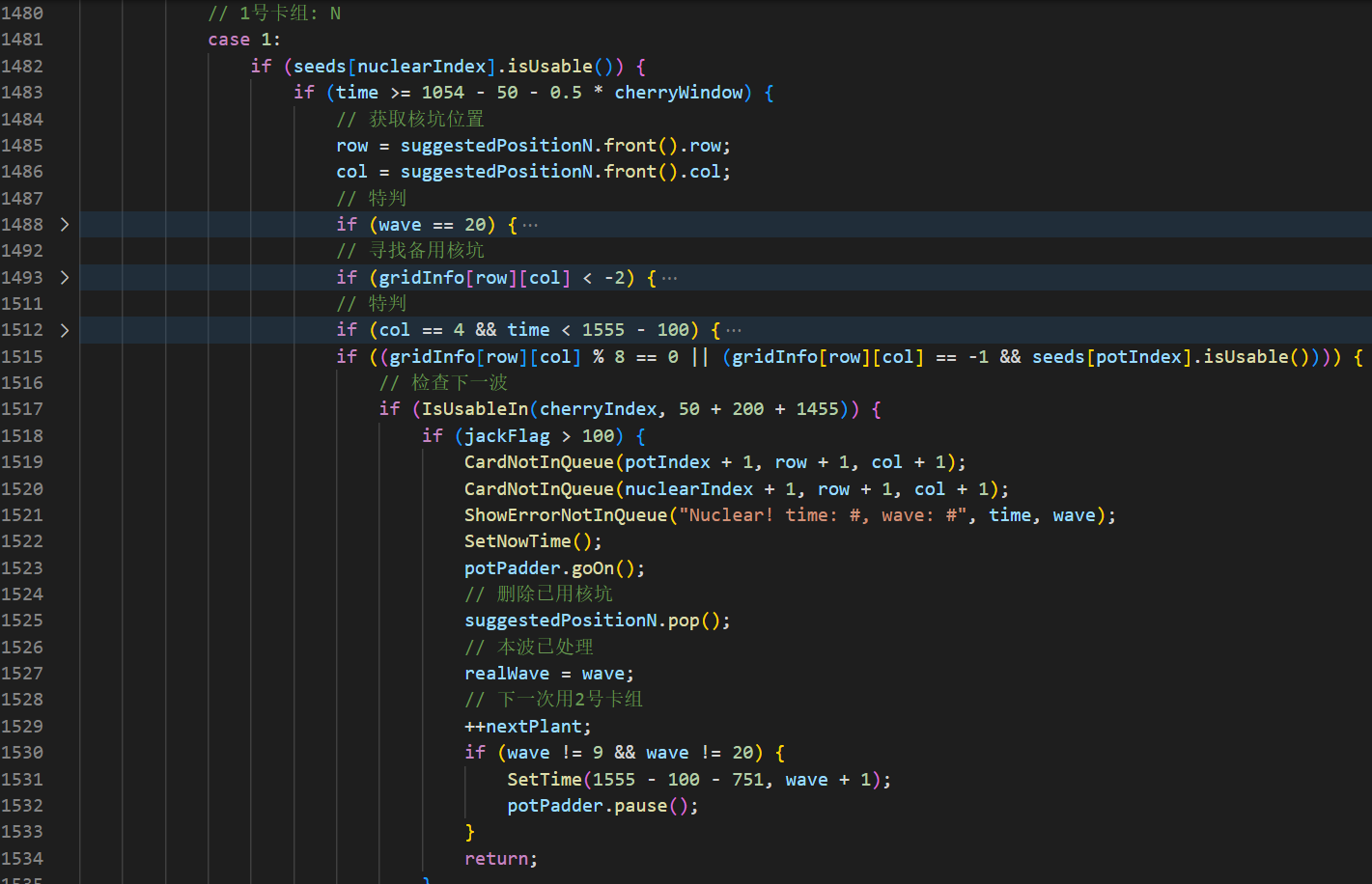
负责变速关的C3change函数仍然是每帧运行的，它主要由3个特判和逐卡主体组成，特判里主要是处理中间波的节奏以及填写核坑、辣椒坑等的位置，这里我们主要看逐卡部分。



这里脚本使用了一个realWave变量来代表上一次处理的波次，以帮助我们判断当前波次是否有处理过。用nextPlant来表示当前应该用的卡组。-1号卡组是第一波用的特殊卡组，用完后设置下一个卡组为0号卡组，即AIN’。

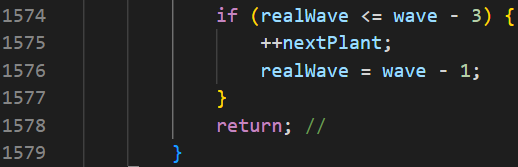


这里是0号卡组，即AIN’的实现。有几个点需要注意：一是由于变速关的波动性，即使是规划好的核坑也可能是不可用的，所以需要备用方案；二是仍然需要判断下一波能否正常衔接，避免循环过载；三是这里把A和IN’拆开处理了，这里只管放A，而IN’由与之前类似的倒计时系统实现。这样做主要是因为放完A之后无论如何都是要放IN’的，所以IN’不应受到各种判断条件的干扰。这个设计可能存在瑕疵，但我自己也没想清楚，就留给读者了。



再来看一个例子，1号卡组N。这里的关键在于核坑读取。在前两种框架中，我们可以根据波次来判断应该调用哪一个核坑，而在这里波次是不确定的，因此我们对核坑的获取，永远是获取“下一组”核坑。这里的核坑操作，相比起之前，是比较灵活多变的。

略过较简单的2号卡组，我们直接来到最后：



这里对整个框架进行了一个兜底。因为尽管我们已经机关算尽，但实际操作中仍会有脚本在一个卡组上卡死的情况出现（基本是因为冰道）。因此，当我们已经有3波或以上没处理的时候，就判定为卡组卡死了，直接强制跳到下一个卡组，来继续运行脚本。

框架介绍部分，完。

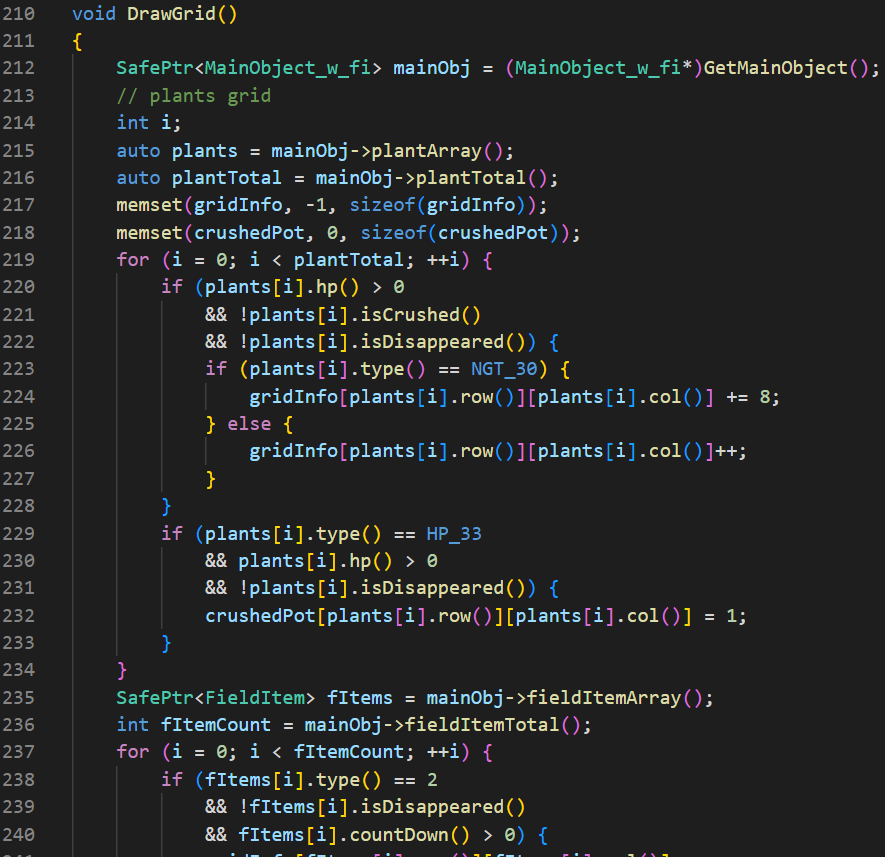
**六、细节与其它功能**

1. 出怪适应



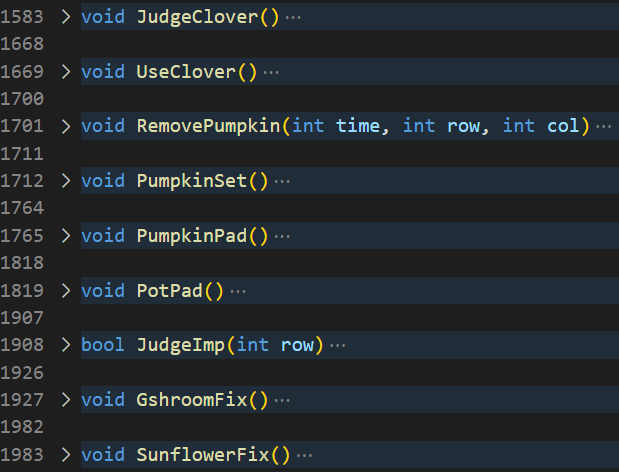
脚本会读取每关的出怪种类，并据此判断选卡和打法。

2. 全场信息



在DrawGrid函数里实现了全场关键信息的实时获取，包括植物种植情况、小丑是否开盒、巨人位置等，以此为放灰烬、补植物等提供帮助。

3. 垫补植物



脚本中有完善的吹气球、补南瓜、补花盆、补曾哥、补向日葵、铲南瓜、垫花盆功能。顺带一提本脚本的花盆控制逻辑是：灰烬优先，垫补其次，吹气球不用花盆。

4. 首尾衔接

挂机脚本不是打一关，或者打一种关就完事的，它需要考虑到某次选卡完成后，下一次选卡cd能否接上的问题，对此需要特别设计收尾，以及可能需要特别设计一些用卡顺序。

**七、缺点与不足**

这个脚本能挂机冲100f，那这是否意味着它已经是个成品了呢？这取决于你的判断标准。可以说是，因为它已经通过了大家公认的检验方式；也可以说不是，因为脚本里还有许多明显的漏洞亟需改进。让我们来看看。

1. 花盆漏洞。在某些情况下可能出现花盆全黑而不能用的情况。出现时机不明，出现原因不明，怀疑可能是avz本身的bug。

2. 三叶草漏洞。在极端情况下可能出现三叶草漏吹气球的情况。出现时机不明，出现原因不明，是谁的问题也不明，大概是我的。

3. 快速关节奏写得极粗糙且随意。建议推重。

4. 慢速关存在许多经验数值，可延展性差。

5. 变速关细节待优化。

**八、总结与展望**

总得来说，写脚本还是比较有意思的，但是修细节就非常没意思。这个脚本本身，我自己是没空+懒得修了，交给大家帮我完善吧，转载注明作者即可。然后这个脚本提出的三种框架，是它的精髓所在，应能抛砖引玉，启发更好的无炮挂机脚本的出现。脚本中出现的一些数据，我并没有说明来源，有些是从网站上查的，还有一些是自己测的。个人感觉写无炮脚本的话，自己测数据的能力也是需要有的。

展望一下，未来是否会出现别的五车场地挂机无炮呢？我觉得短期内还是难说，但最终总是能的。回看这个脚本，里面满是为容错而做出的妥协，这又是否必要呢？我希望不仅能出现别的五车场地无炮挂机，而且能出现打得更精确、脚本更简明的那种。非五车场地中还有一个NE场景，至今很少有人开发无炮挂机脚本。这其实是一个比较奇怪的现象。至少从我今天的视角看起来，NE的无炮挂机应该是要比ME简单不少的，甚至可能不用这些复杂的框架也能写个大概，不知道有没有人愿意尝试一下呢？DE和RE就是两个非常难的场景了，其中以RE为最。不知道有生之年能不能见到RE的无炮挂机脚本。

**九、致谢**

感谢@向量cwl 开发AvZ脚本框架。这个脚本框架用起来真的很方便，而且还性能好无延迟，可以说是无形中为我省去了巨量时间。

感谢@雪漠北 设计核聚变无炮阵型。如果没有这样像我这种手残也能打好的五车场地无炮，我可能也不会想到去写这样一个挂机脚本。

感谢@刚刚雨纷纷 的交流和支持。没有你的话这个脚本不会有现在的曝光度，我也不会写这个贴子。

感谢@风若轻雨 的挂机测试。这东西的测试我也做过，是件很消耗时间和耐心的事情。感谢你耐心测完100f，证明了脚本的能力。

感谢@虚奴Nhs 脚本及本文作者。