MicroRts第三次实验报告

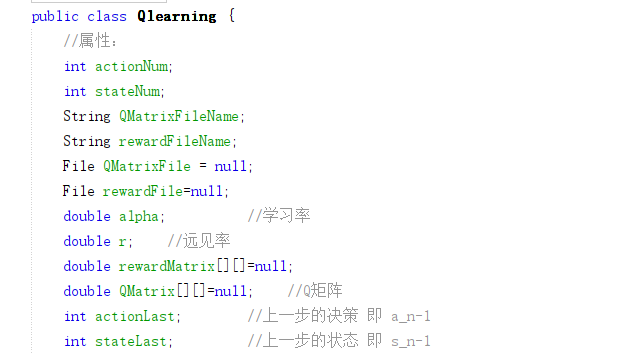
09017423 杨彬 09017421 胡世杰

前言：这几周的工作总体概括就是在之前的基础上，基本实现了AI的决策学习能力，能够根据局势作出相应选择，并在几轮后能战胜本不能战胜的敌方AI。

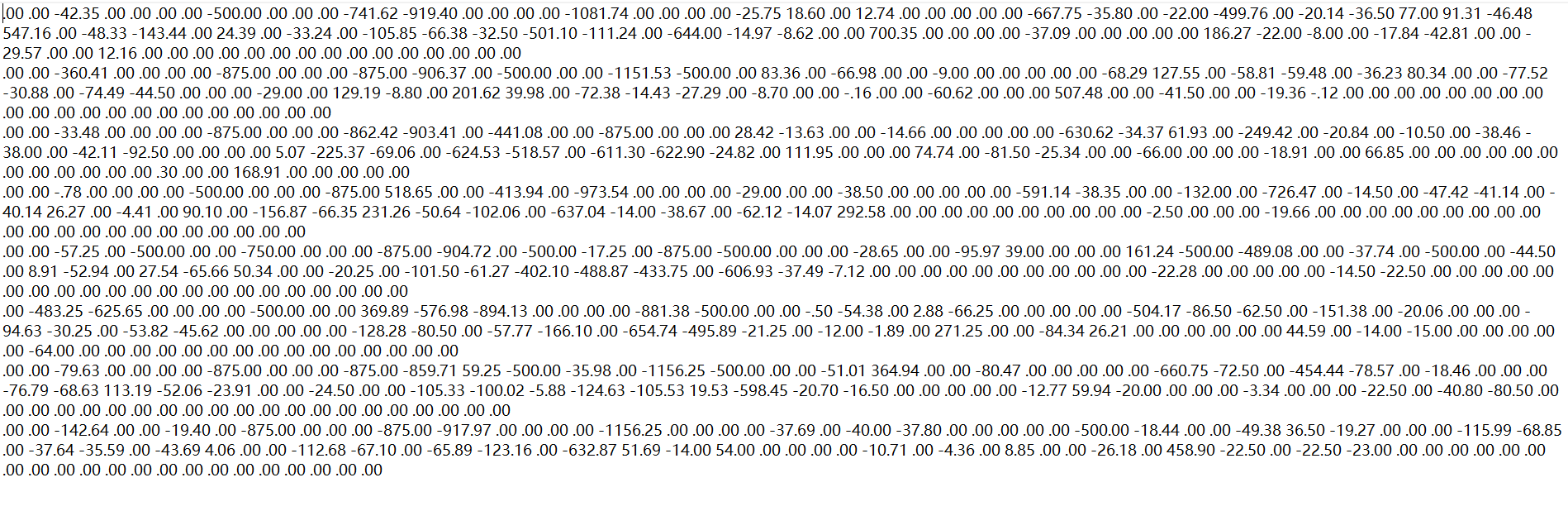
**PART I 已完成的工作**

1. **构建了一个Q-learning类**

作用：从文件中读取现有Q矩阵，根据矩阵进行决策，再根据决策后的战场情势更新原Q矩阵并输出到文件。Q-learning类包含了使用Q-learning算法进行训练所需要的接口，包括更新，决策，学习等等。

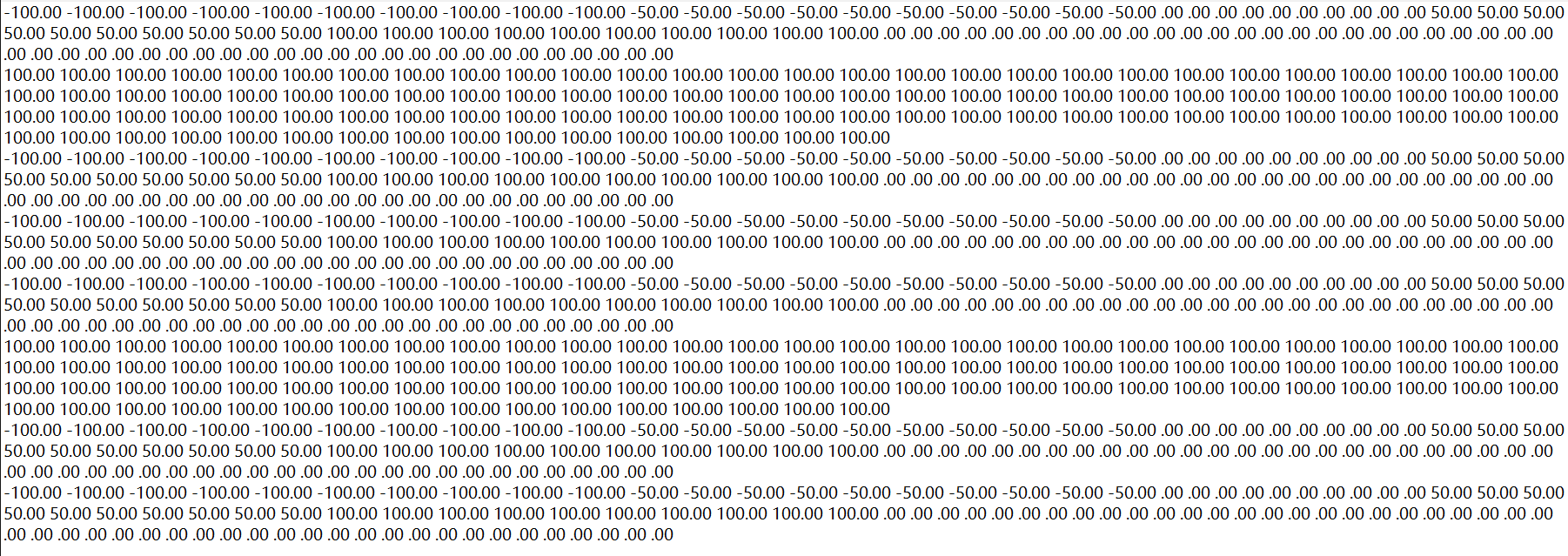
 Qlearning 类的成员变量和函数

我们根据我方目前的优劣局势（大劣势、劣势、平衡、优势、大优势）、敌方的攻守状态和敌方主要兵种（worker、light、ranged、heavy、worker-light、worker-ranged、worker-heavy、light-ranged、light-heavy、heavy-ranged、light-ranged-heavy）给出了5×2×10总计100种state，再给出攻守判断和兵种判断（四种）给出2×4共8种决策，形成一个8×100的Q矩阵。



（经过训练后的Q矩阵 8\*100）

一开始，我们还构建了一个reward矩阵（与Q矩阵相同大小、表示Q矩阵在某个局势下采取某个决策带来的收益值），作用为作为部分权重来更新Q矩阵。我们的初步设想是人工给定reward的值，再通过大量训练获取一个最终的Q矩阵，但经过训练发现Q矩阵的更新并未对局势带来好的转变，只有在随机选择决策时可能带来优势。检查后发现reward矩阵的赋值出现了根本性的错误，同时发现人工给定reward值有两点不足：一是很难做到全面和精准；二是与人工决策的区别不是很大，失去了学习的意义。



（原本的reward矩阵）

于是我们决定采用动态的方式获取reward值来更新Q矩阵，而获取的方式我们粗略大致决定采用使reward值等于决策后局势战力差与决策前战力差之差（战力差是之前的内容，即根据对比双方的兵种数量、距敌方基地距离的综合性判断得出我方与敌方的战力总和之差，是用来判断战场情势的一部分指标），方法的正确性不太确定，但是我们目前能得到的比较可靠的数值（reward值的给定就是Q-learning的一大难点所在），经过之后的训练也可大致确定没有大方向上的错误。

1. **使用Q矩阵进行训练学习**

在决定了上述的基础后，我们准备进行AI的训练。

我们的Q-learning的训练是每隔一段固定的时间进行一次学习，在这一次学习的时候我们获得了当前的state，我们将上一次决策的state和战力值以及上一步决策采取的action保存下来，根据当前的state我们计算出上一步的决策的reward,于是我们先更新上一步决策的Q矩阵，然后再进行决策。如此循环反复。我们采用了固定的决策周期，根据观察，对阵进攻型AI时较快的结束时间为600~700左右（根据左下角的times），对阵防守型较慢的结束时间为2200+, 故我们取100作为决策周期（过长过短都不太好）。



（getAction里的决策模块）

在训练初期，我们将没有任何决策的我方AI与自带的AI进行战斗训练，在几轮后问题暴露，与进攻型AI战斗时，除了在第一轮可能在随机状态下持续选择了较好的决策胜利后，之后的每一轮都是败，而且是惨败，并且最多的决策选择为worker（根据经验，light和range是兵种中较强的，也是最该被选择的）。之后的几十轮对战，变动的是开局选择建造的兵种，而不变的是始终惨败的结果。在审视了训练后得到的Q矩阵，我们发现在开局初期的状态下8种决策的值均为负值，而每次的选择都是从中挑取最不差的进行决策，而即使人工给定其中某个决策极高值后，在几轮训练后仍旧会更新为负值。

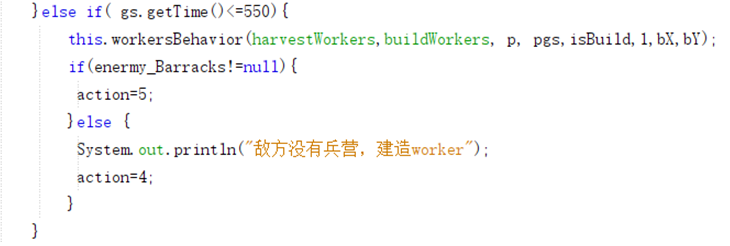
经过思考，上述情形的出现应该源于reward值的给定：

1. 前期建造兵营耗时长，而建造兵种须在兵营建完后才能进行，这导致选择建造worker可以带来稳定且高于其他决策的战力增长（即使worker实际很弱），这使得worker决策的reward值虚高，故经常选择worker决策。

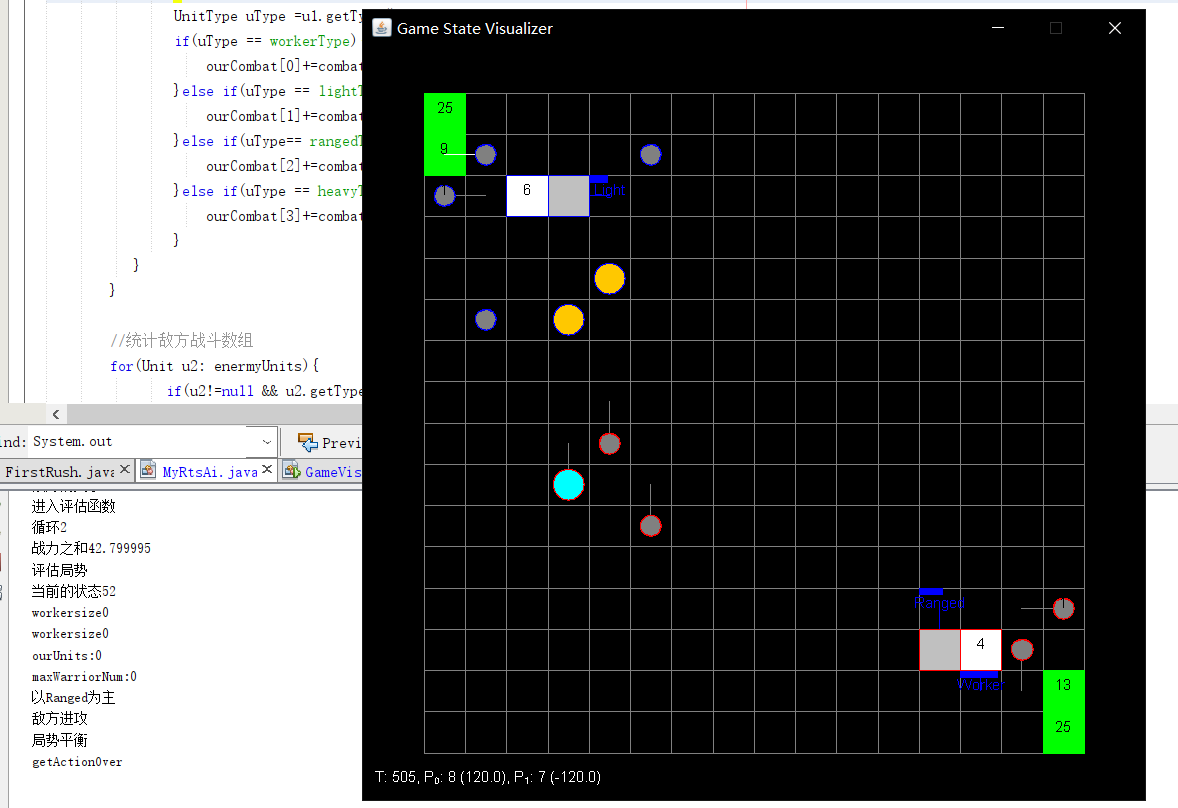
2、本游戏的节奏过快，在开局即使做对决策，第二步走错便可能导致失败，从而使得决策难以学习。

总结：此游戏的胜负90%甚至可以说完全取决于前面几百times的行为（在16×16的地图下，更小的地图节奏更快，8×8中worker rush就是王者），而在前期，对于AI来说几乎没有任何的决策参考，让AI学习比较困难，同时前期的一个决策失误会直接导致gameover,比如在对阵workerRush的时候，如果我们一开始就建造兵营，那么一定会被打死，但是在对阵别的AI的时候，不尽快建造兵营，又会导致前期的劣势很大。问题在于我们无法判断地面的AI采用的是什么策略（在地图大小比较小的时候有其明显），所以我们决定给在前面的550times采用人工智慧，后面再交托给AI。我们给定的前期为：

在敌方建造出兵营前，我方开始建造兵营和持续采用worker defense，在兵营建造完后，采取light defense策略直到550times。在确定前期决策后，AI的胜率得到了极大的提高。

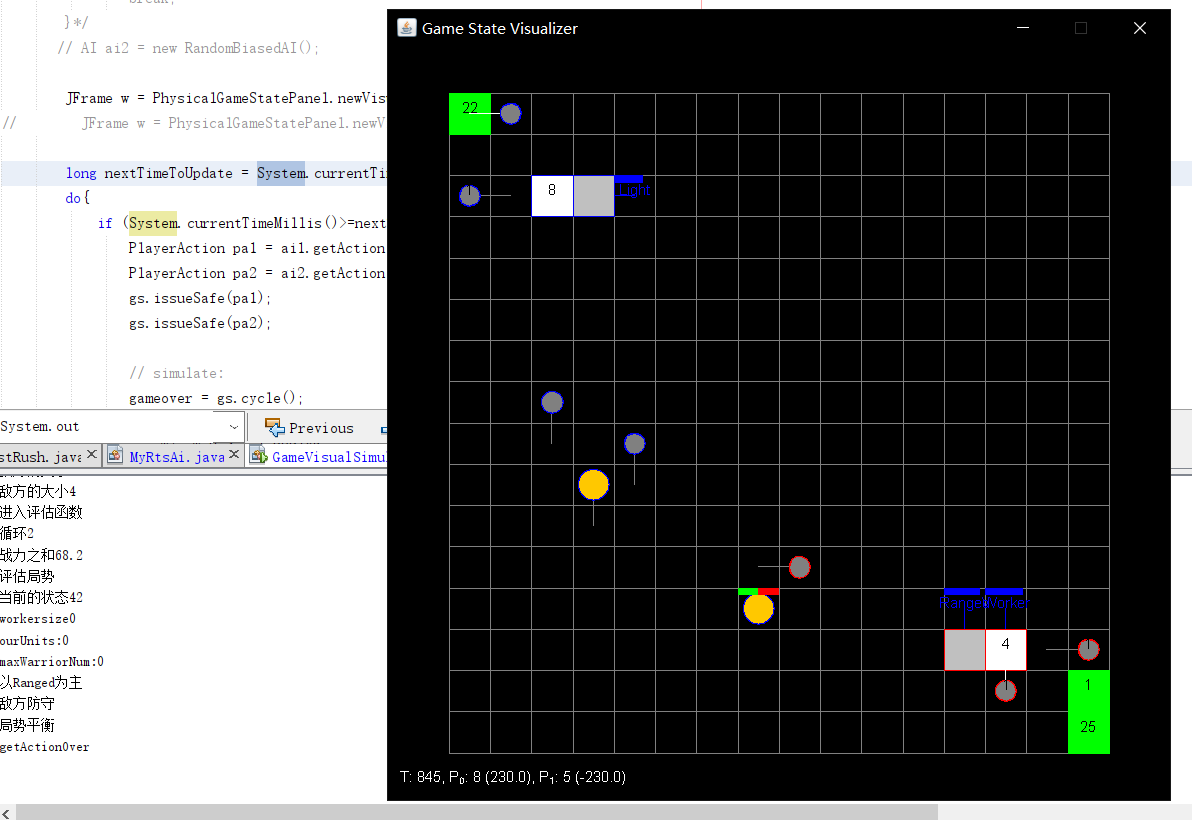


（在游戏前期（即time<=550）的时候采用固定的人工智慧）

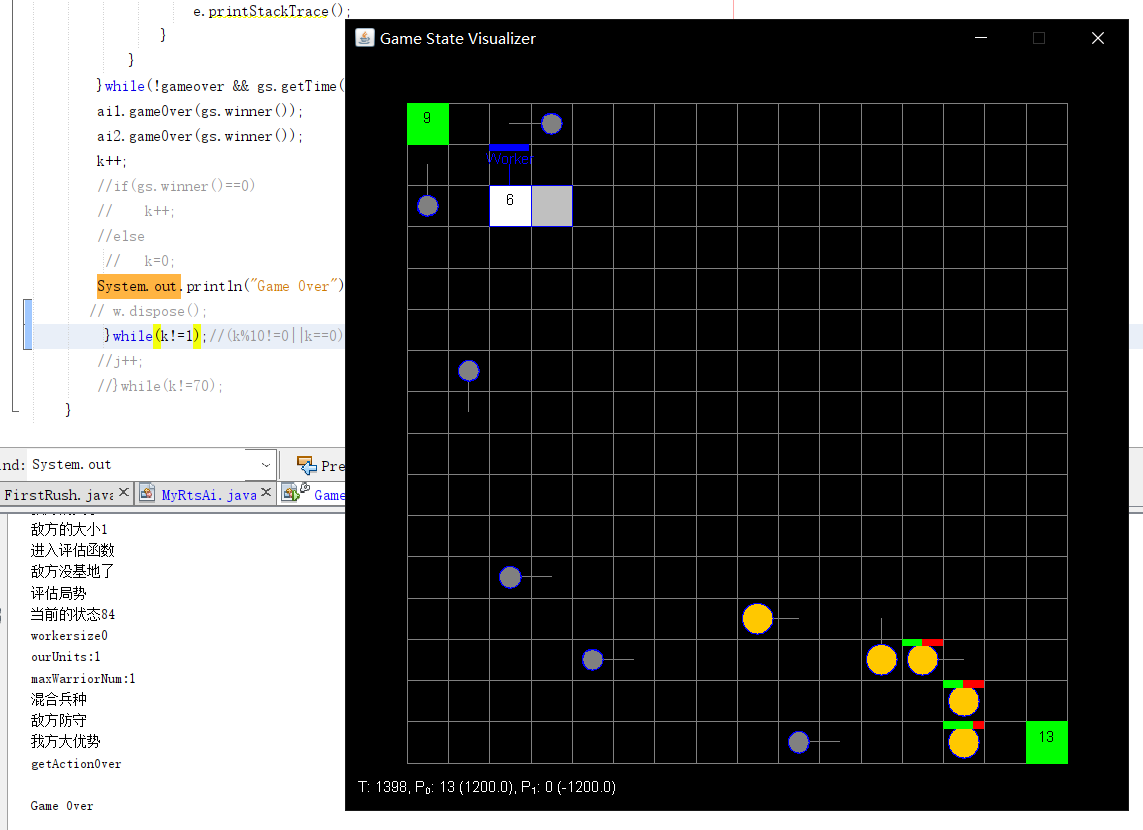
 （前期局势）

1. **对阵不同AI进行循环训练**

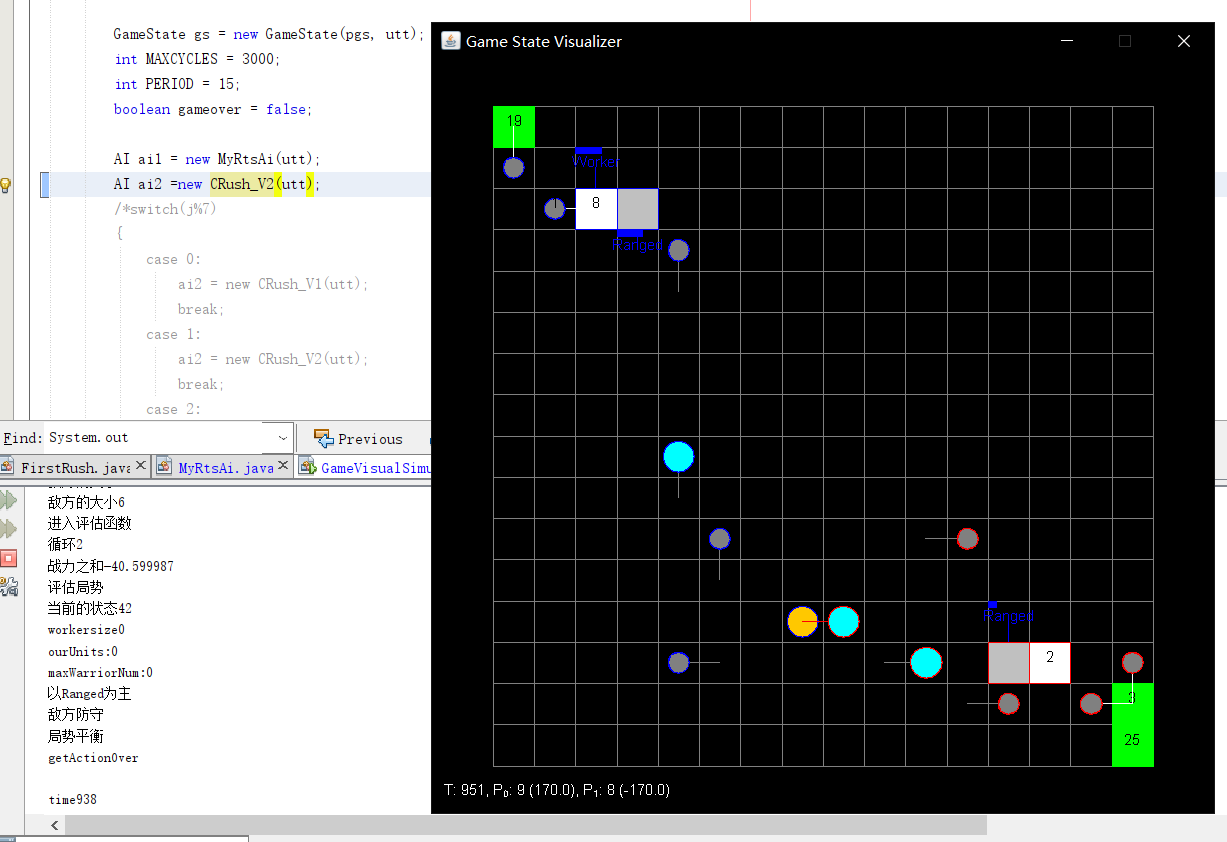
1 我们在对战进行的文件下进行了更改，使得我们的AI可以循环的与别的AI进行对战（连续战胜10次同个AI才能与下一个AI进行对战否则连胜次数清零，总计n个AI，则总的连胜次数至少要达到10n）。在经过不断调整参数（比如time step 训练周期），在参数合适的情况下，我们发现一开始我们的AI可能打不过对方的AI但是经过数局（正常是十几局）之后我们的AI已经能够战胜大部分该游戏自带的AI了，比如比较强力的 Crush\_V1, Crush\_V2, lightRush,lightDefense 等等。（下面给出对战部分AI的结果，我方位于左上角）



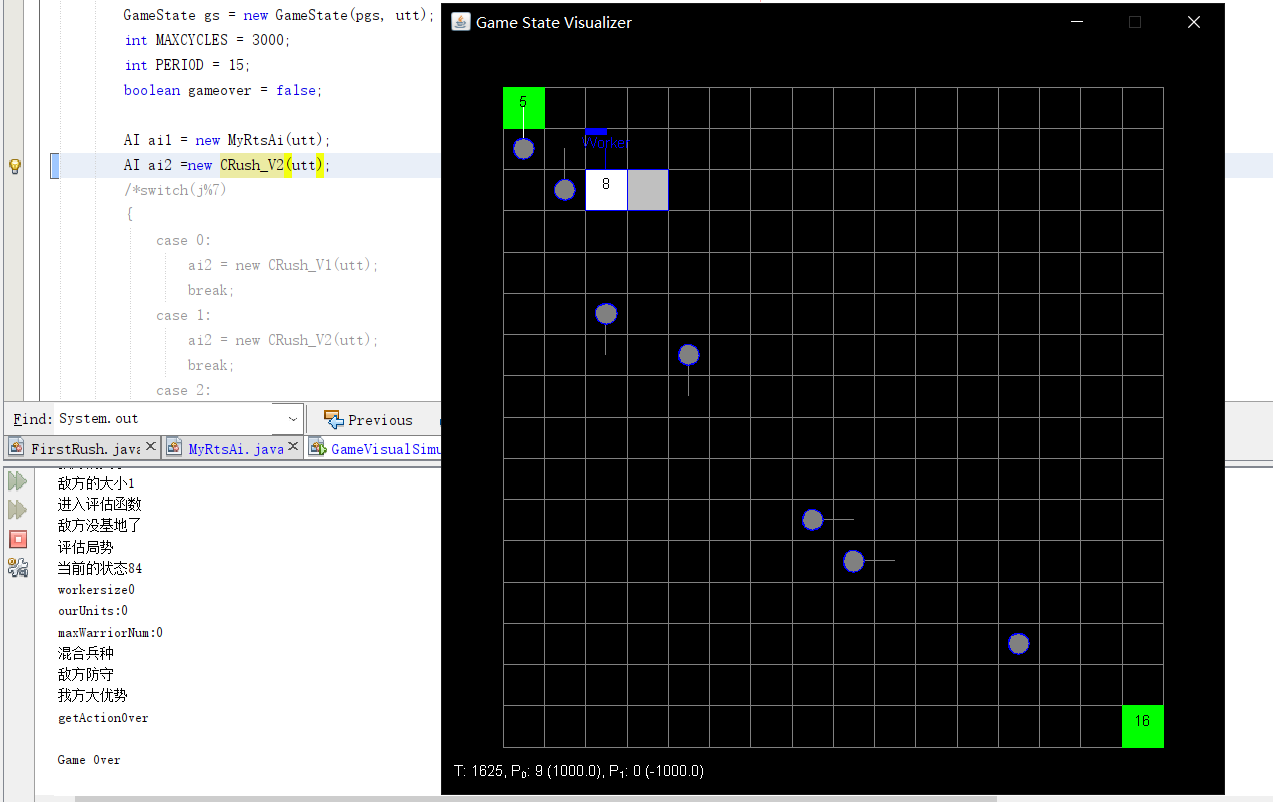
( VS Crush\_V1 )



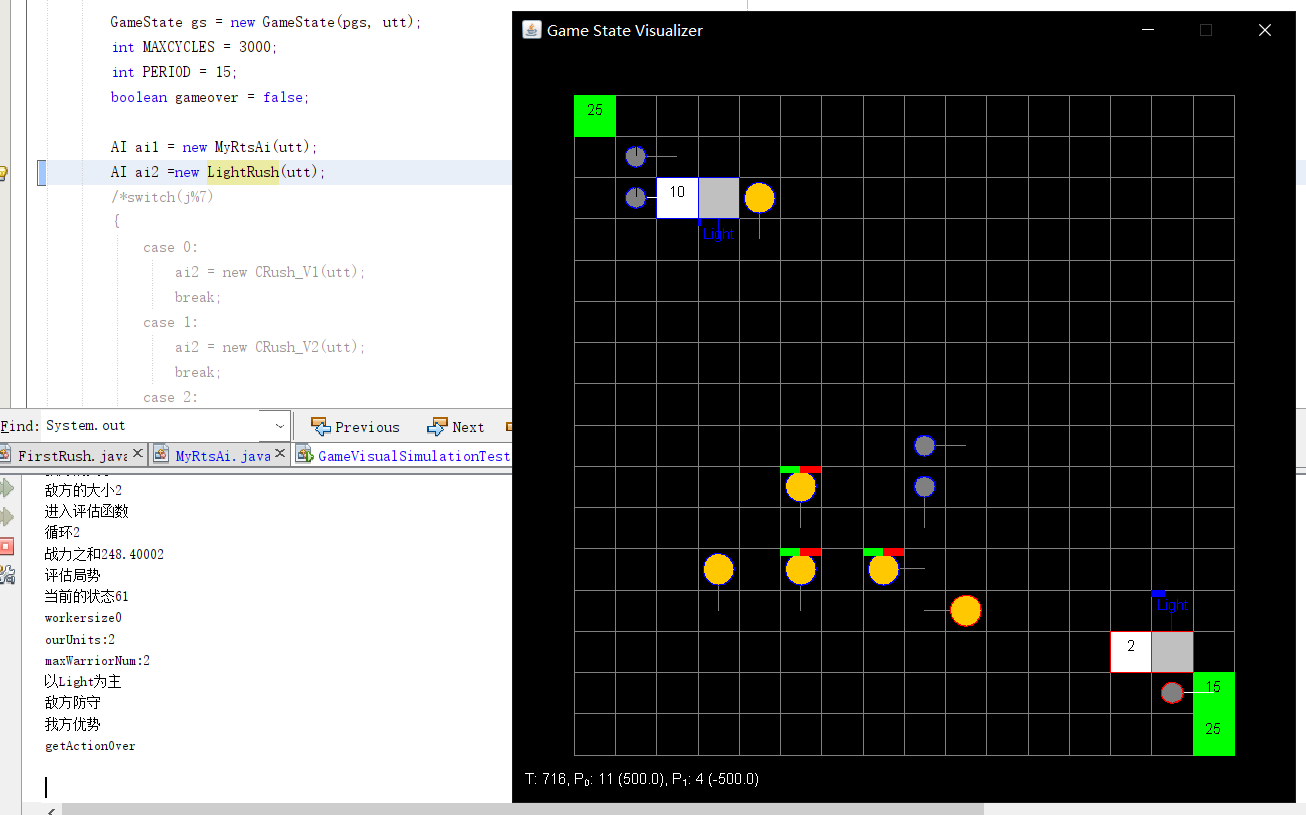
( VS Crush\_V1 )



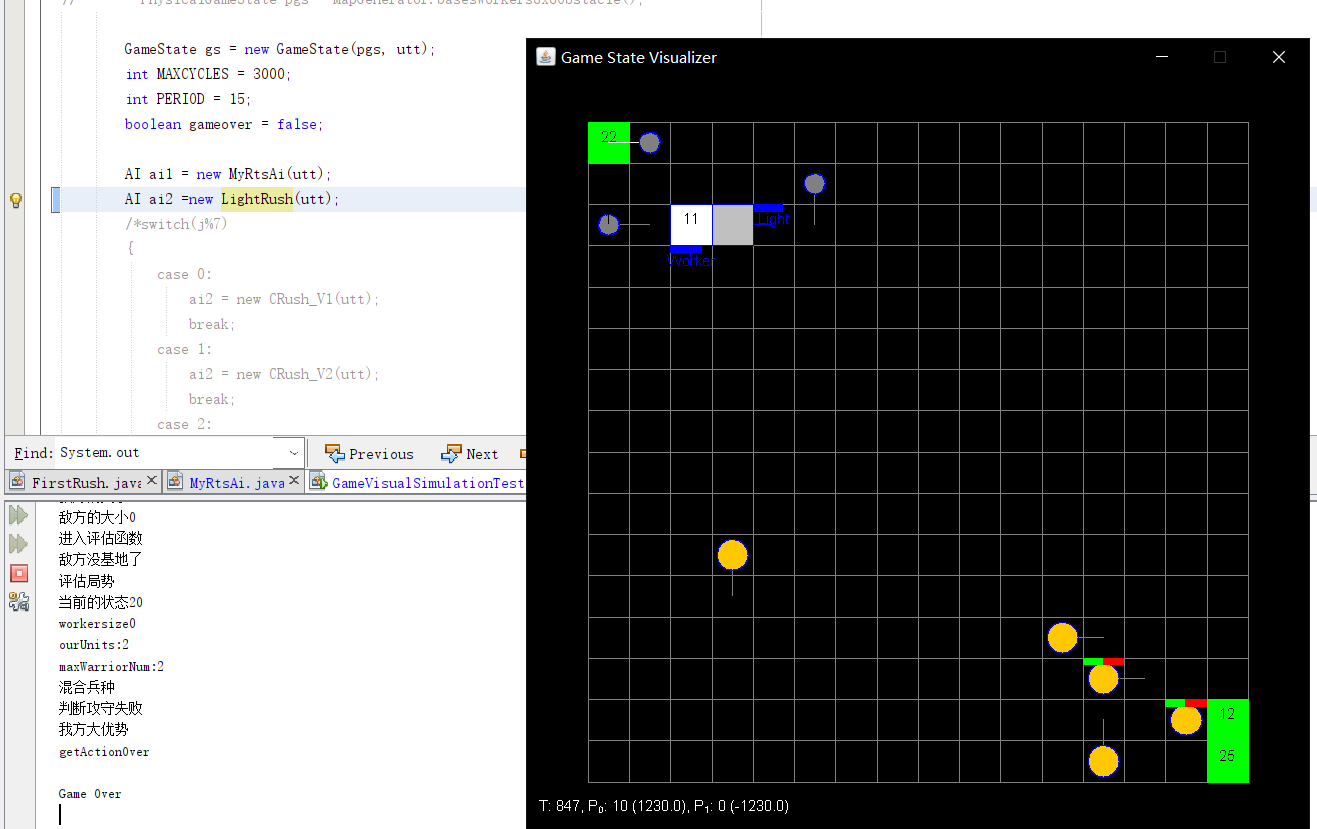
（VS Crush\_V2）



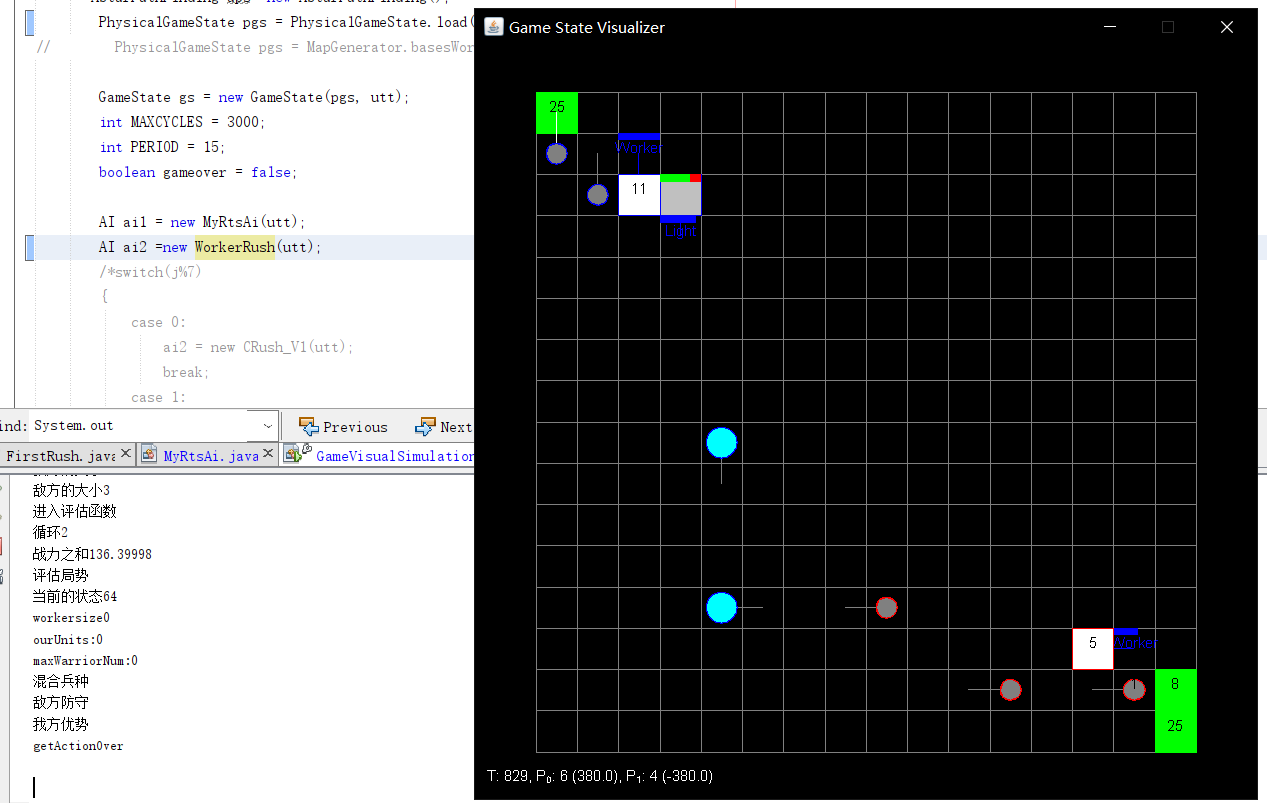
（VS Crush\_V2）



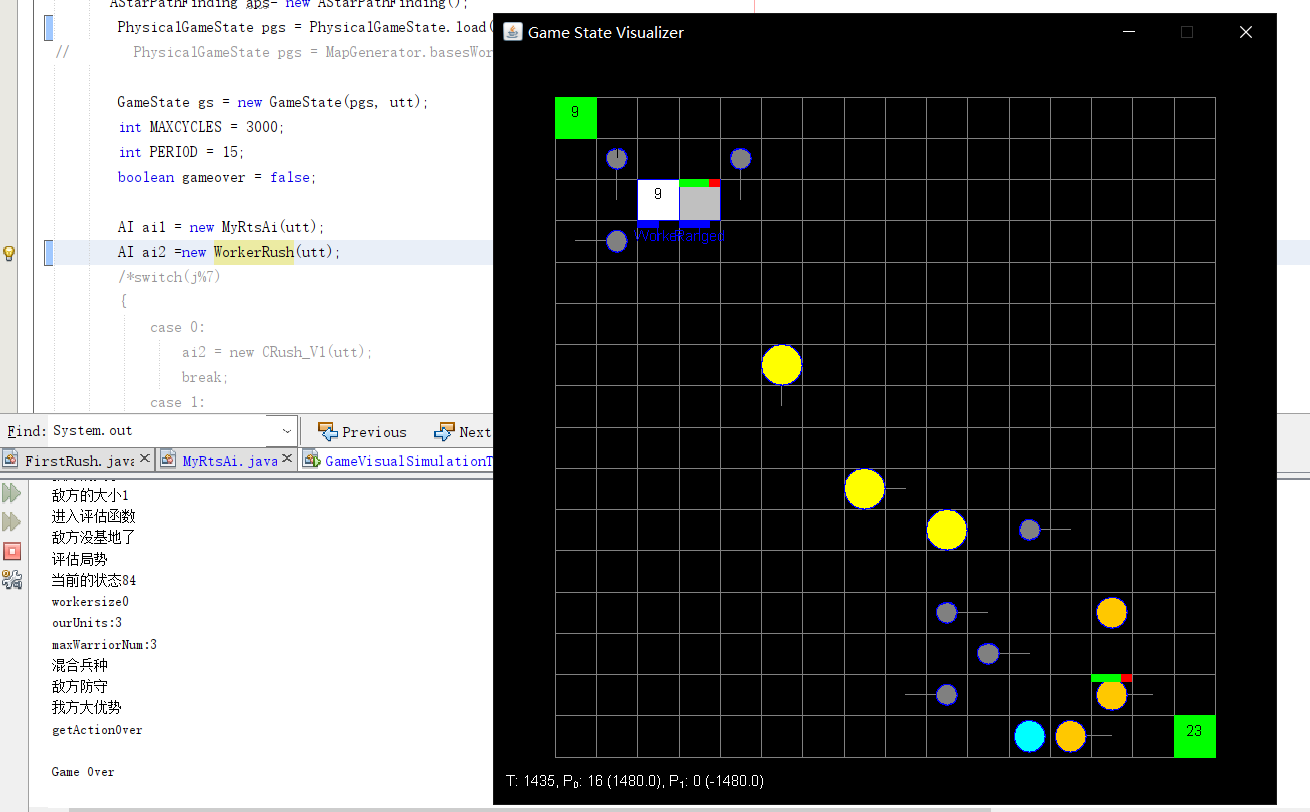
(VS LightRush)



(VS LightRush)



(VS WorkerRush)



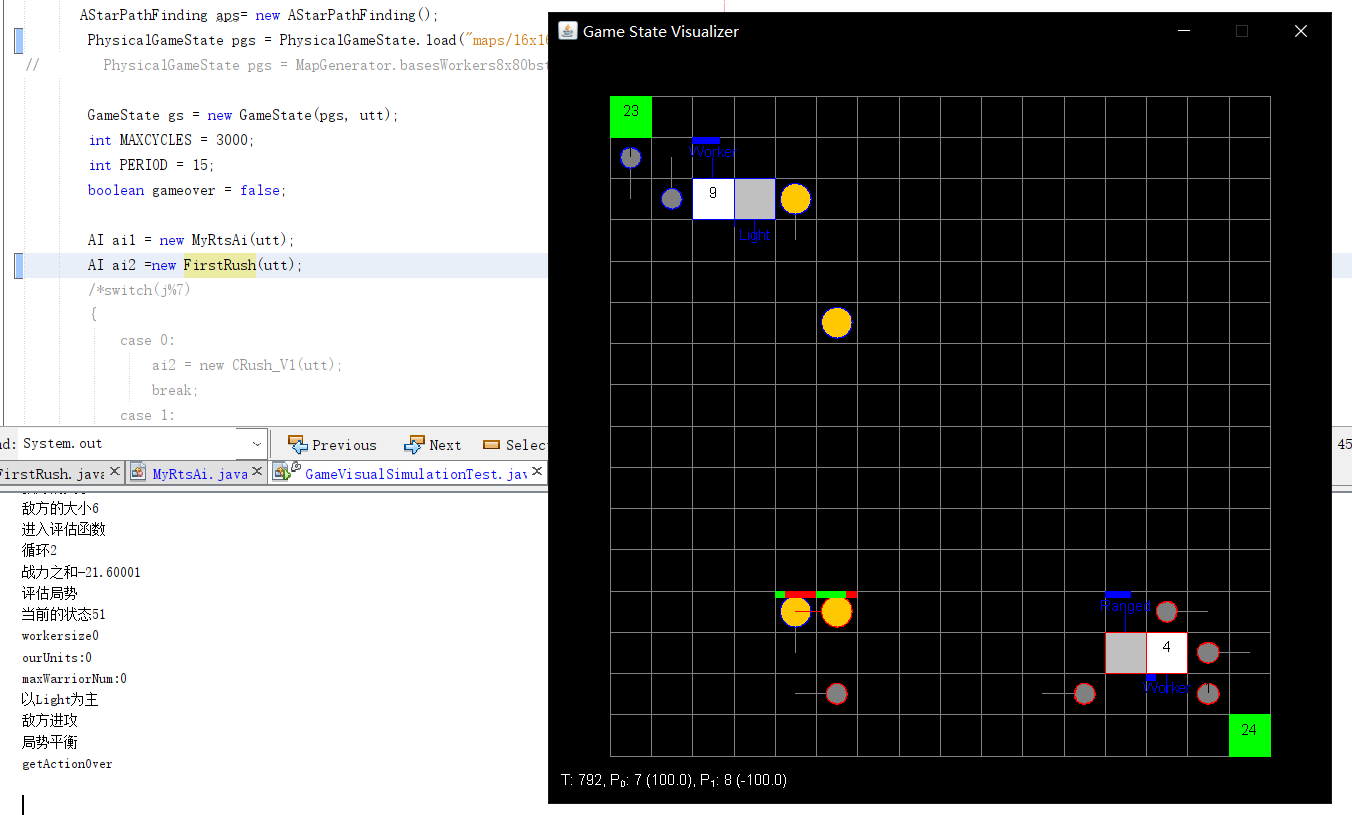
(VS WorkerRush)

可以看到目前我们的AI能够战胜大部分的自带的AI，并且在不同的情况下，AI采取的策略不同。

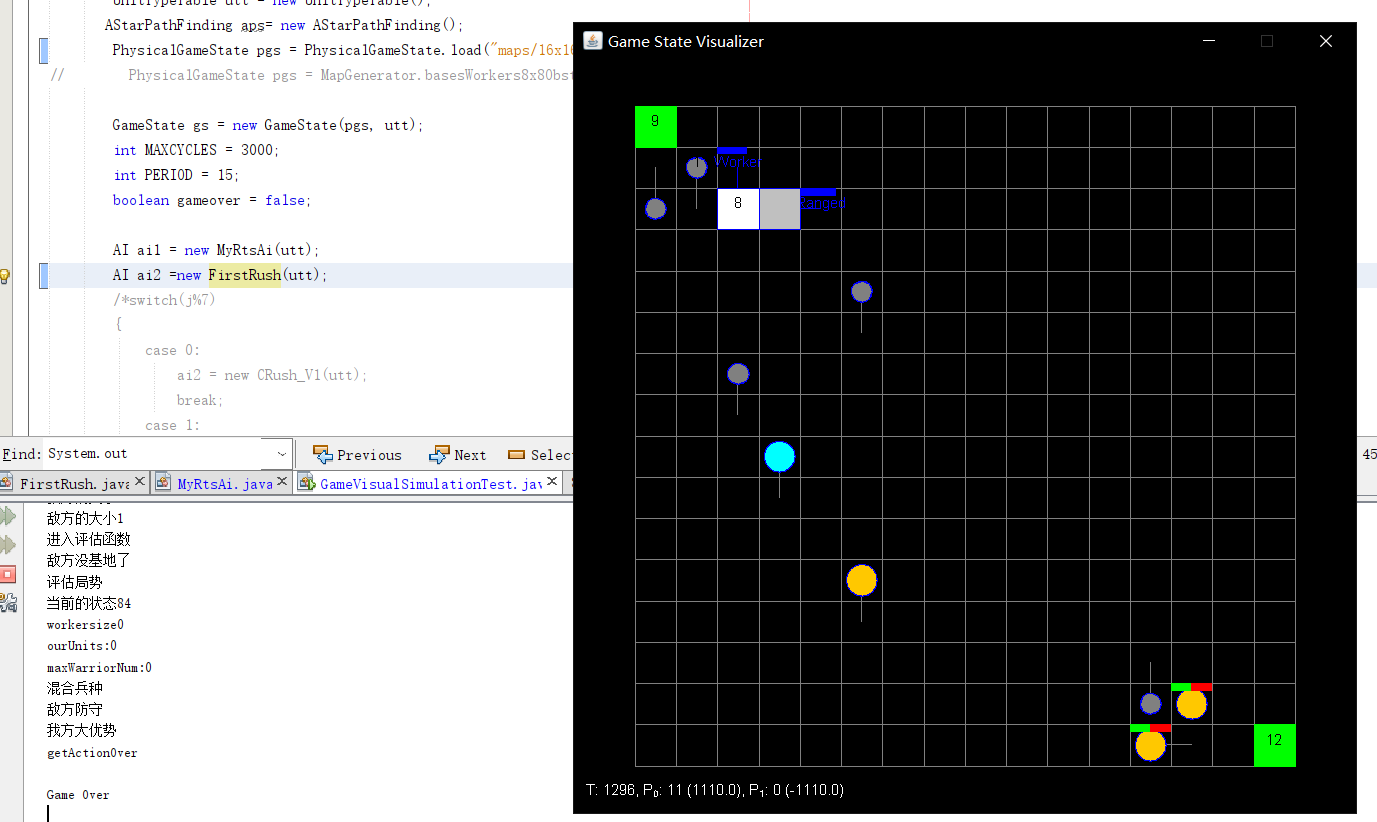
除此之外，我们还进行过一次尝试，我们让自己的AI打自己的AI，即左右互博，对战

一段时间之后，我们再让我们的AI取对战电脑的AI，我们发现，大部分的AI是能够打赢的，这从一定程度上说明了我们的学习还是有一定的效果的。

在对战电脑AI之后，我们要来了班级某个同学的AI，并和其对战，一开始是输的（因为我们的AI只和电脑的AI大过，训练样本不够大）但是经过循环对战之后，我们的AI已经能赢了） 下面给出对战的结果（左上方是我方）



（VS 同学的AI）



（VS 同学的AI）

**PART II存在的问题和未来展望**

**一、存在的问题。**

Ⅰ、开始训练后，发现对阵第一个AI花了七八分钟（那时没有注意能调整周期来改变对战速度，此处为原速）才能战胜，第二个花费了十分钟，后面的较快，但经过一个循环后，重新对战到第一个AI却又重新惨败，且花费二三十分钟仍不能战胜。

经过观察，问题大致可能为三点：

1. 基础的行为执行的太差。
2. 状态数不够导致矛盾的出现。即状态存在矛盾。在状态k下对阵AI1采取a方案才能赢，而对阵AI2是在相同的状态k下采取b才能赢。
3. 防御决策的reward值反馈无法体现。若双方均采用defense，则结果必然是一段时间的积攒后发起的总攻才能体现，但根据我们的reward值给定，无法体现，所以会出现在好几个time\_step前建造的ranged在这个time带来的优势算在了现在的heavy rush决策上。

Ⅱ、我们发现不管是对阵别人的AI还是左右互博，我们的训练容易陷入一种死循环，也就是训练无法再前进了，貌似已经得到了最优解，但是仍然是输的，我们重复对战了50局左右，每一句的结果甚至是过程都是一样的，虽然游戏本身的机制存在问题，但这也说明了，我们的训练仍然存在问题。

Ⅲ、无法准确的给出reward值，由于整个游戏是不断动态进行的，与走迷宫，或者寻找最优路径这种传统的强化学习应用场景不同，而reward又具有延迟性，我们很难得到在什么时候进行评估，要延迟多久，如何给出准确的reward，前面提到，我们的训练周期是固定的，本次训练的reward在下一次训练的时候根据下一次训练的state计算得出，这体现了reward的延迟性，但是固定的训练周期存在很大的问题，比如我在这一时刻选择建造Hearvy,但在下一次决策的时候，战力差确实得到了增长，也就是reward变大了，但是这时候确实由于之前建造的ranged消灭对方大量worker导致的与上次决策的heavy并没有什么关系，换句话说，ranged的功劳被算在了heavy的头上，这明显是不科学的，其根本原因我们认为有三：

1、采用固定的训练周期，导致不灵活

2、reward的给定有问题，不能准确反映收益

3、我们的决策是局部的决策（第二次报告提到，我们只决策兵营出什么兵种，和采用进攻还是防守的策略，并没有细化到对每个单位进行决策）但是我们的reward确实全局的reward。

Ⅳ、我们AI的基础行为，比如说进攻不够完善，具体表先在1、在采取防御时，我方兵种被攻击后却无反应2 经过多次训练比对发现，我方的range的实力远不如某些AI，按照我们最基础的兵种战力比对，range应该是克制heavy，甚至是无损风筝，而我们的range只会站桩输出，更新后应做到走位+攻击。

**二、未来的改进和展望**

1 优化reward函数，让reward的评定更加科学

对于前面提到的reward的问题，我们认为第三点，局部决策和全局reward的矛盾按照目前的决策和学习的框架，我们还无法解决，也比较难解决，我们觉得这是这个框架啊本身的缺陷。但是我们能够完善reward函数，比如对交战（双方有兵力的损失）和非交战（双方都在屯兵）状态采用不同的reward策略，在交战状态下，reward的主要标准应该是杀敌数或者是战损比。而在非交战状态下，由于兵种存在着克制关系，我们这时候应该根据双方兵种的组成和兵力来给出reward，以解决之前提到的防御reward的问题。

2采用动态的训练周期

前面提到了，固定的训练周期会导致功劳归属出现问题，也就是reward的计算会产生问题，因此训练周期应该是动态的，我们初步决定，训练周期的时间为，建造一个单位的时间+该决策单位到达对方战场中心的时间（对方坐标平均值），这样能基本保证在这段时间内，这个被决策的单位实际参与了战斗，在一定程度上解决了问题，当然对于攻击防守，也应该采取不同的计算方式，归根结底就是让reward函数更加准确。（其实reward函数有些类似与深度学习里的损失函数）

3完善我们的基础操作

例如

（1）在采取防御时，我方兵种被攻击后却无反应，更新后应做到反击。

（2）完善ranged的进攻函数，让我们的ranged也能够风筝别人

（3）丰富action动作，比如编写不同的进攻策略，防御策略的函数，让AI的决策有更多的选择（目前是有8个action有点少了）

4 动态改变学习率

在训练过程中，我们发现了有可能一个训练的比较好的Q矩阵，在几局失败之后被打破了，因此我们考虑参考深度学习的思想，对于学习率 alpha，采用动态的方式，随着寻来你次数的增加，学习率应该减少，在矩阵已经比较好的时候，在学习的时候能够更多的保留之前学到的知识，这应该是比较科学的做法。



（Q-learning 的核心函数，和学习率）