MicroRts第四次实验报告

09017423 杨彬 09017421 胡世杰

前言：最后几周的工作，我们针对上一次提出的问题，比如QLearning的核心reward函数、timeStep(学习、决策间隔)，基础Ai等，对我们之前的Ai进行了优化和完善。增加了state，但是由于时间和游戏平衡性原因，新版的AI并没有得到充分训练，Q矩阵还没有收敛、导致新版的Ai打不过旧版的Ai，因此rts比赛我们提交了旧版的、完善过的Ai，但是在这篇实验报告中，我们仍将详细介绍我们对QLearning的改善，给出并分析部分实验结果

**PART I 对QLearning的改进**

**一 改进了reward的获取**

正如我们在报告3中所述，Qlearning的核心在于reward的获取，事实上reward也是强化学习区别于其它学习算法的主要特征，我们之前的Ai的主要问题也主要是reward获取太过僵化、不够合理，新版的AI的核心正式对reward的改进。改进reward的思路如下:

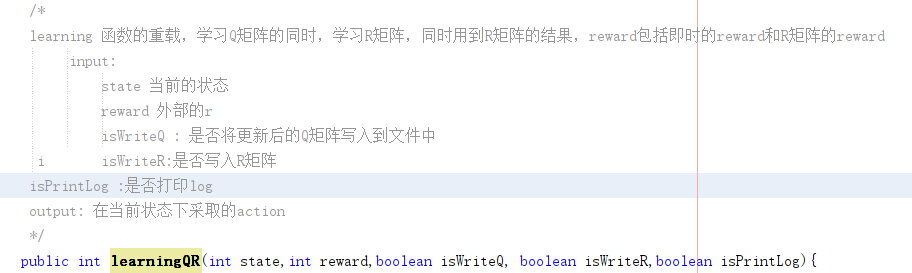
参考了Qlearning的核心是训练Q矩阵，从另一个角度想想？我们为什么不能同时训练R矩阵呢？一开始Qlearning学习公式[1]里的reward是从固定的reward矩阵里获得的，但是这样意味着R矩阵的值需要人工给定，而人工给定reward结合rts游戏特殊的性质，人工给定reward可能导致reward不够合理的问题；后来我们采用了动态的reward值(正如实验报告3所述),reward值是前后两次决策阶段的战力值之差，这样我们就不用人工给定reward值了，但是这又带来一个问题，由于我们是人工划分state（也就是说我们人工将连续的状态离散化），在同一个state下，得到的reward也可能会有很大的差距，为了解决这一问题，我们的解决方案如下：

1. 我们采用了和训练Q矩阵相同的思路——训练R矩阵。训练R矩阵的公式如下： （1）,其中 R是R矩阵（reward矩阵）我们称之为经验池，它表示之前所有的学习过程所获得经验的总结，i是action,j是state，reward是第n次决策的reward值，alpha是R矩阵的学习率alpha值越大就越容易抛弃之前学到的reward。每一次决策前的更新，我们获取的reward除了用来更新Q矩阵之外，我们还用来更新R矩阵，这样每一次更新的reward都会不断地被积累下来，我们有理由相信，如果状态划分地足够科学（换句话说状态之间的区分度较大，而同状态（state）之间的区分度较小），那么当训练次数足够大之后，R矩阵应该同Q矩阵一样收敛到某一个值的。
2. 每一次更新获取reward的公式如下 ：

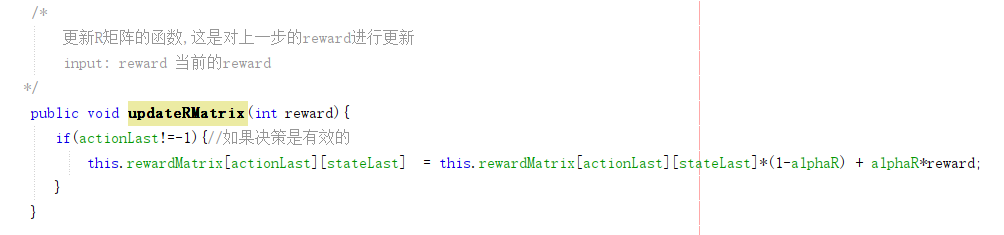
在这里 reward表示最终用来更新的reward值，即（1）式中的reward,R是reward矩阵，而weight是权系数，r表示通过战场局势获得的即时的reward表示，上式的含义是用来更新reward由两部分组成 1是过去学到的reward知识，即经验池的值，2是新的知识，即由战场局势获得的reward。上面的式子综合了R矩阵和即使的r，获得的reward就是这一次更新最终的reward，用来更新R矩阵，和Q矩阵。

3 为了更科学地获取战场的即时r，解决在报告3中我们所说的攻击防御reward的问题，我们交战状态和非交战状态，采用不同的策略，如果决策后到更新的过程中我们判断双方AI并未交战（判断是否交战，先计算双方所有单位的X坐标和Y坐标的平均值，然后计算 ,如果值小于一定的阈值就判定双方处于交战状态）,如果是非交战状态，我们根据兵种的克制关系，综合对方的主要兵种和我方上一步做出的决策给出固定的r，如果是交战状态，我们用以下公式获取固定的r :，level表示的是局势等级（同之前的战力值的含义类似）,这么做除了让reward值给定更加科学之外，还可以将reward值限定在一定的范围之内，防止过大或者过小，采用这种方法我们能更加科学合理地根据战场局势的前后变化获取reward。

综合1、2 、3reward给定更加地科学，既不需要人工给定的reward，又能够同时考虑到以前的情形和当前的局势，经过对战的观察我们也发现，reward的值比较好地符合预期，即我觉得Ai采取的策略是不错的时候，reward给的一般是正的，而且值比较高。反之也比较合理。对应到代码中：



（新增learningQR来同时学习Q矩阵和R矩阵）



（更新R矩阵）



（计算战场r的函数）

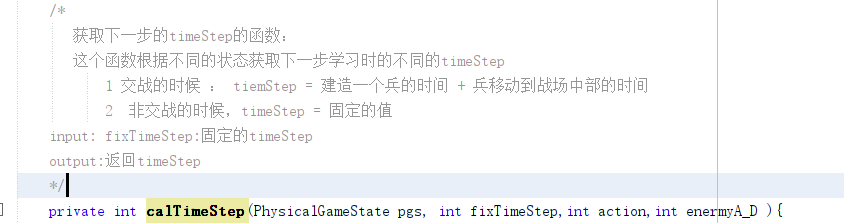


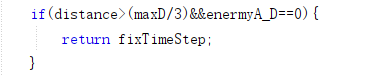
（reward矩阵的值）

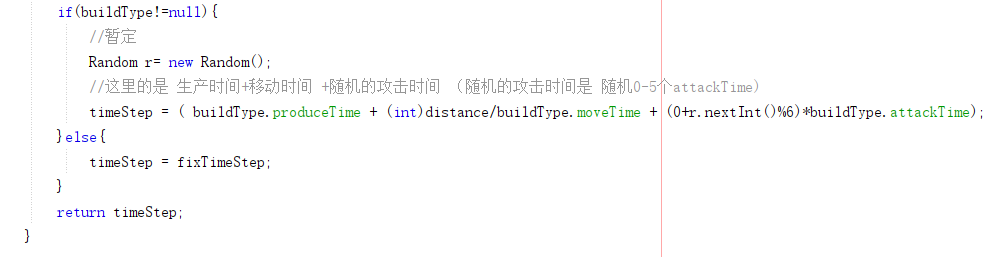


**二 改进了timeStep**

我们还对QLearning的训练周期进行了改进，区别于旧版本的AI，我们在新版本的AI中采用了动态的训练周期，正如我们在实验报告三中所提到的，如果采用固定的训练周期的话，会导致,reward无法正确地反映到上一步的reward中，因为可能出现这种情况，这一次获取的reward事实上是上上次的action(比如这一次我们选择出了light，但是兵确实由上上次出的ranged打的，然后reward算到了ligt的头上)，而采用动态的timeStep虽然不能够从根本上解决这个问题，却能够尽可能地缓解这个问题。我们考虑采用这种方式获取reward: 1 如果在这一次决策的时候，双方处于未交战状态，那么我们认为，之后双方交战的可能性也不大，因此timeStep采用固定的100，如果是交战状态，timeStep的公式如下 : , buildTime是选择的action对应的建造单位的建造时间，distance是从兵营位置，移动到对方战场中心（也就是对方的横纵坐标平均值的位置）的距离，moveTime 是对应单位的移动速度，rand(1,5)是1-5的随机整数，而attackTime是进行一次攻击所用的时间。上式的含义是，决策的周期应该是建造一个兵种，并且让兵种移动到交战区域并且做出贡献的时间，显然采用这种策略获取动态的timeStep,能够让reward反映的更加科学。对应到代码中：



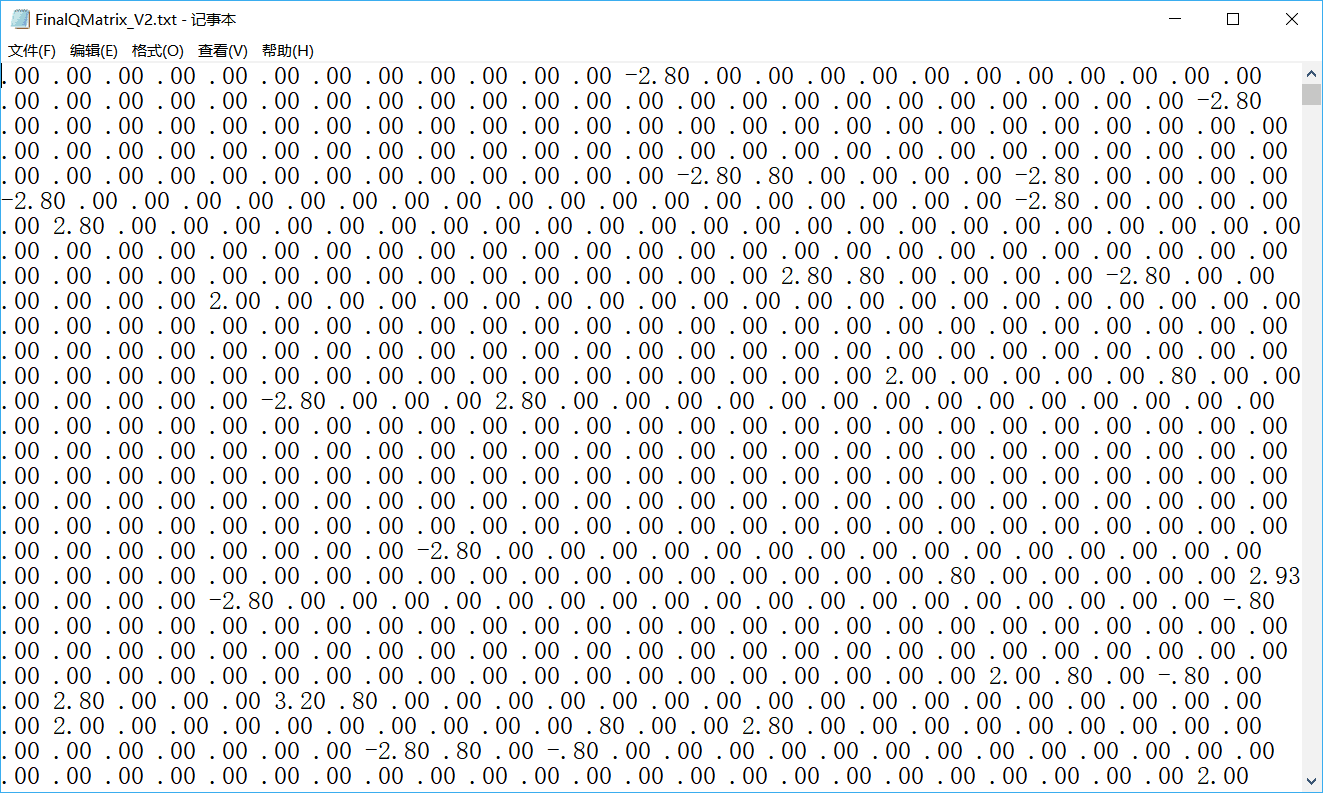




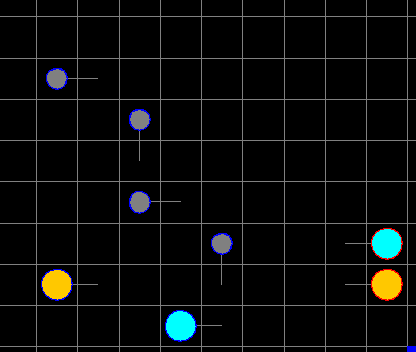
（设置calTimeStep计算 timeStep，以针对不同的情况采用不同的timeStep计算方式）

**三 增加了更多的状态**

正如我们在报告三中所述，状态过少将会导致矛盾地出现，由于我们采用的QLearning，状态是离散的，这是QLearning这一经典的强化学习算法的固有特点，我们无法改变，因此直观、直接的做法就是，增加更多的状态，在上一个版本的AI中我们值考虑了 1局势（5个） 2 敌方的攻防状态(2个) 3敌方主要兵种（10个） 一共是 5\*2\*10 = 100个 state。在这一个版本的AI，我们除了考虑对方的战场信息，我们还考虑了己方的战场信息，并且并且将局势状态增加到了10个,在这一个版本中 1局势（9） 2 敌方攻防（2）3我方攻防（2）4敌方主要兵种（5） 5我方主要兵种（5） 一共有 9\*2\*2\*5\*5 = 3600个state。State的大量增加，带来的好处是决策更加灵活，比如我多次看到了己方出现（light+ranged）的兵力配合（light+ranged很强，因为light可以在前面给ranged当挡箭牌）碾压对面，这就是综合考虑己方的state的结果，但这也带来了一个问题,state太多，收敛太慢，大多都还是0，由于时间原因以及临近考试周，我们并未完成对ai的调试，这也是一个遗憾吧。

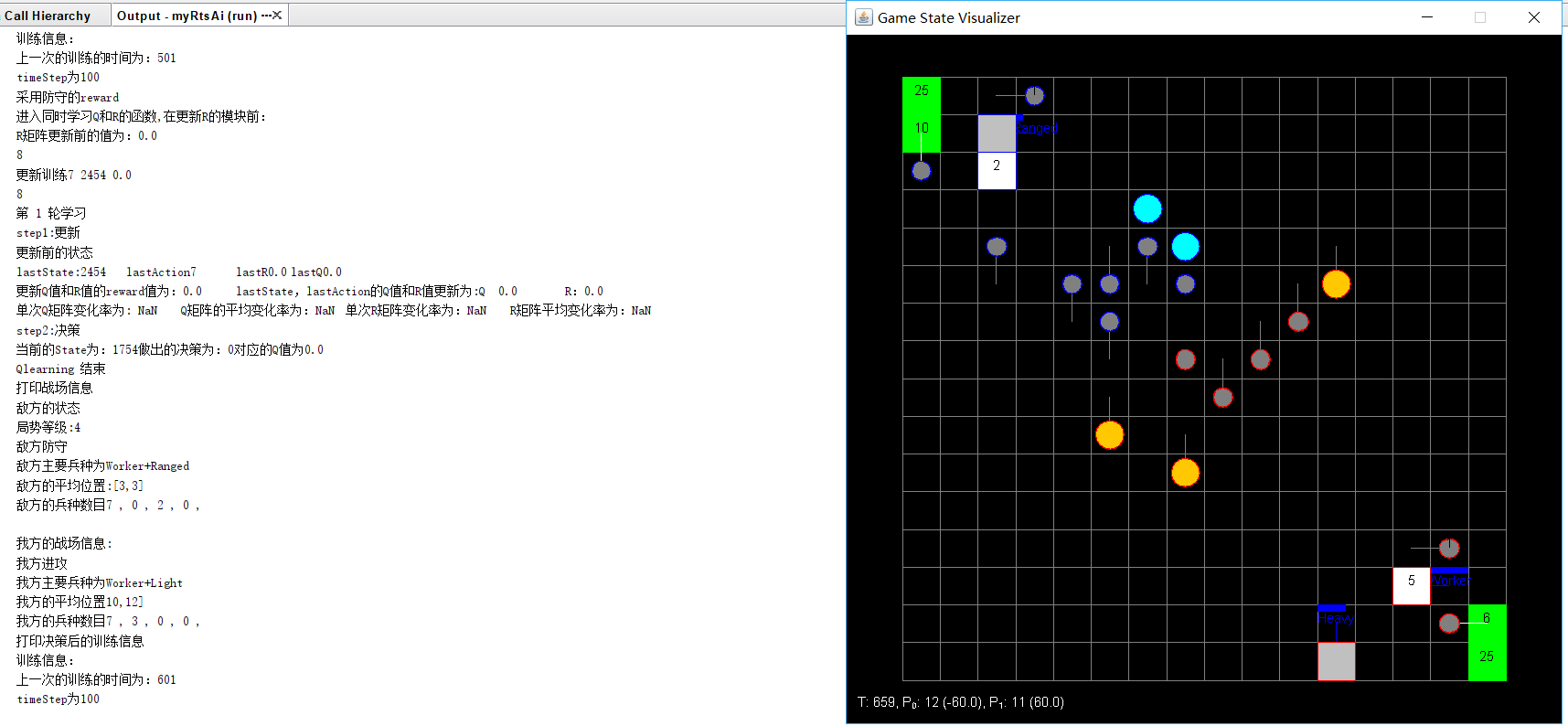


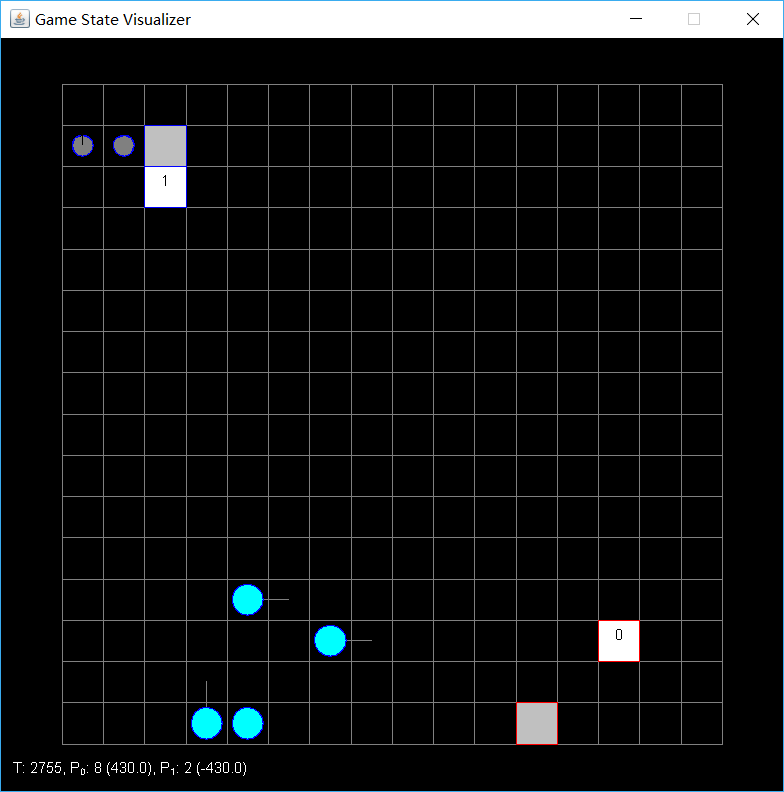
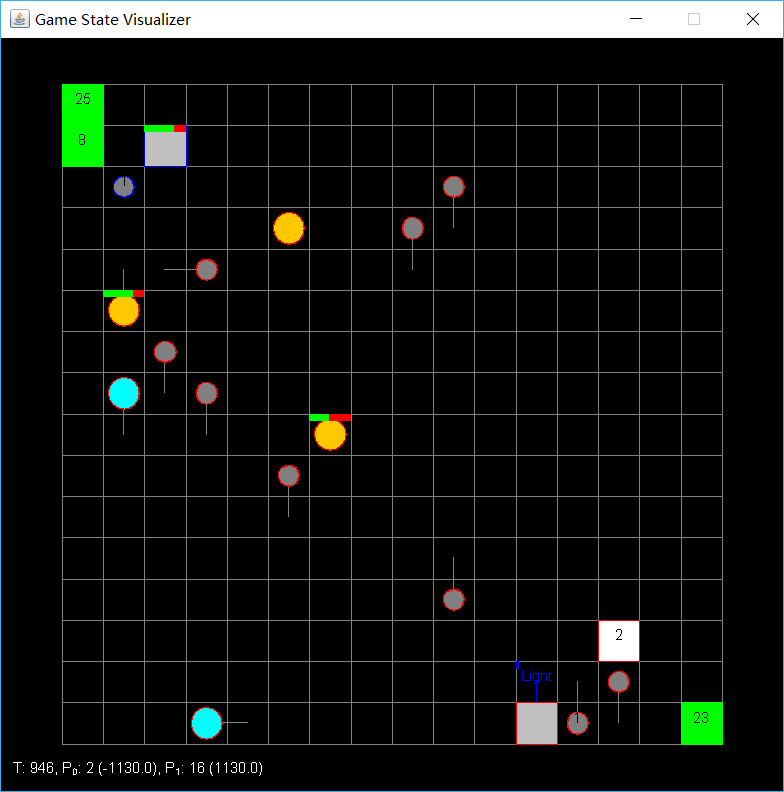
除此之外我们还优化了基础AI,比如防御的函数，比如把hearvy从出兵表中删去（游戏平衡形的原因heavy实在太垃圾了）比如增加了action，让生产light的同时也能够生产worker等等。

 **（ranged和light的配合）**

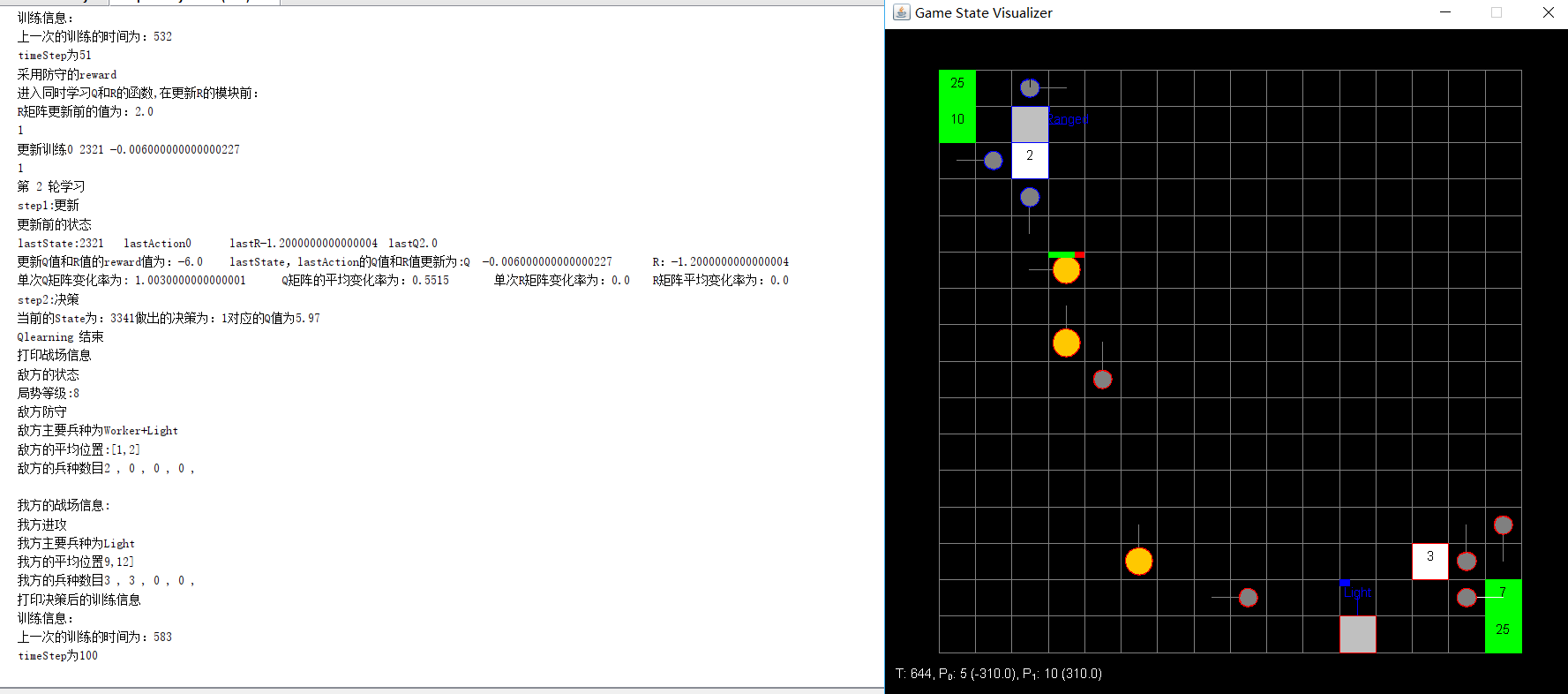
**PART II 一些对战结果和部分训练日志**

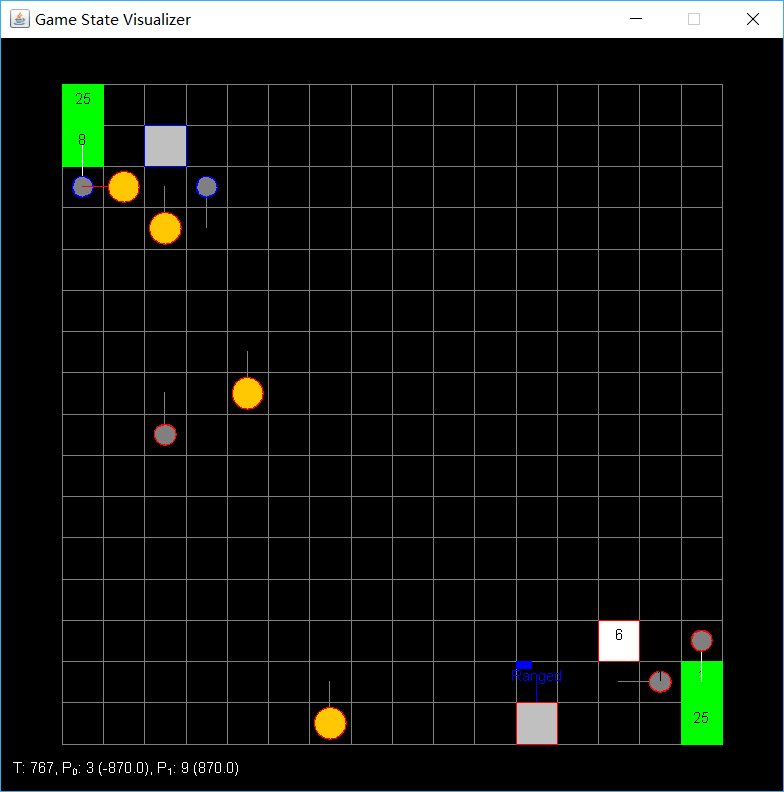
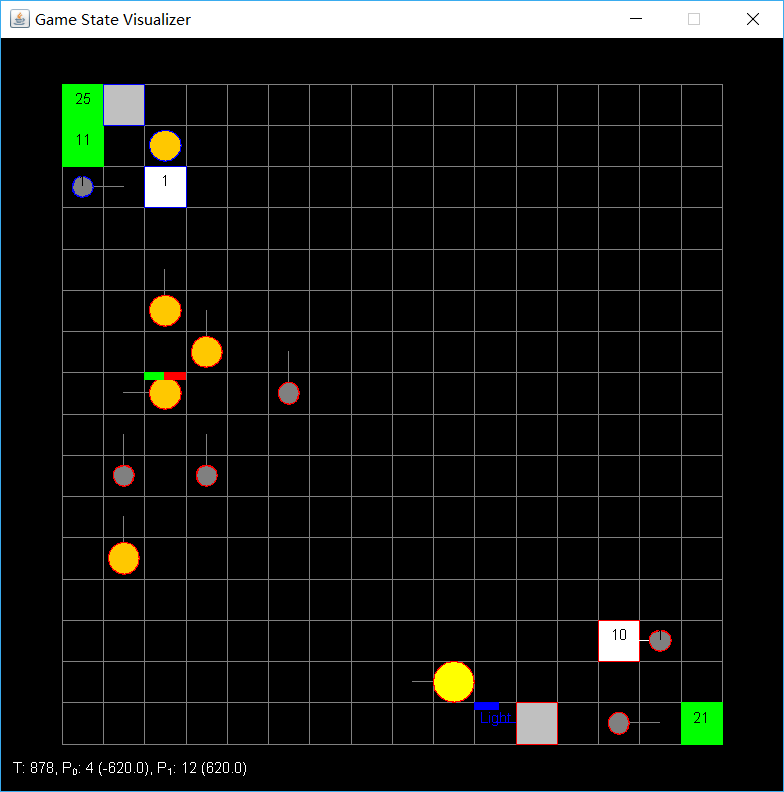
1. 右下 对战CRush\_V2



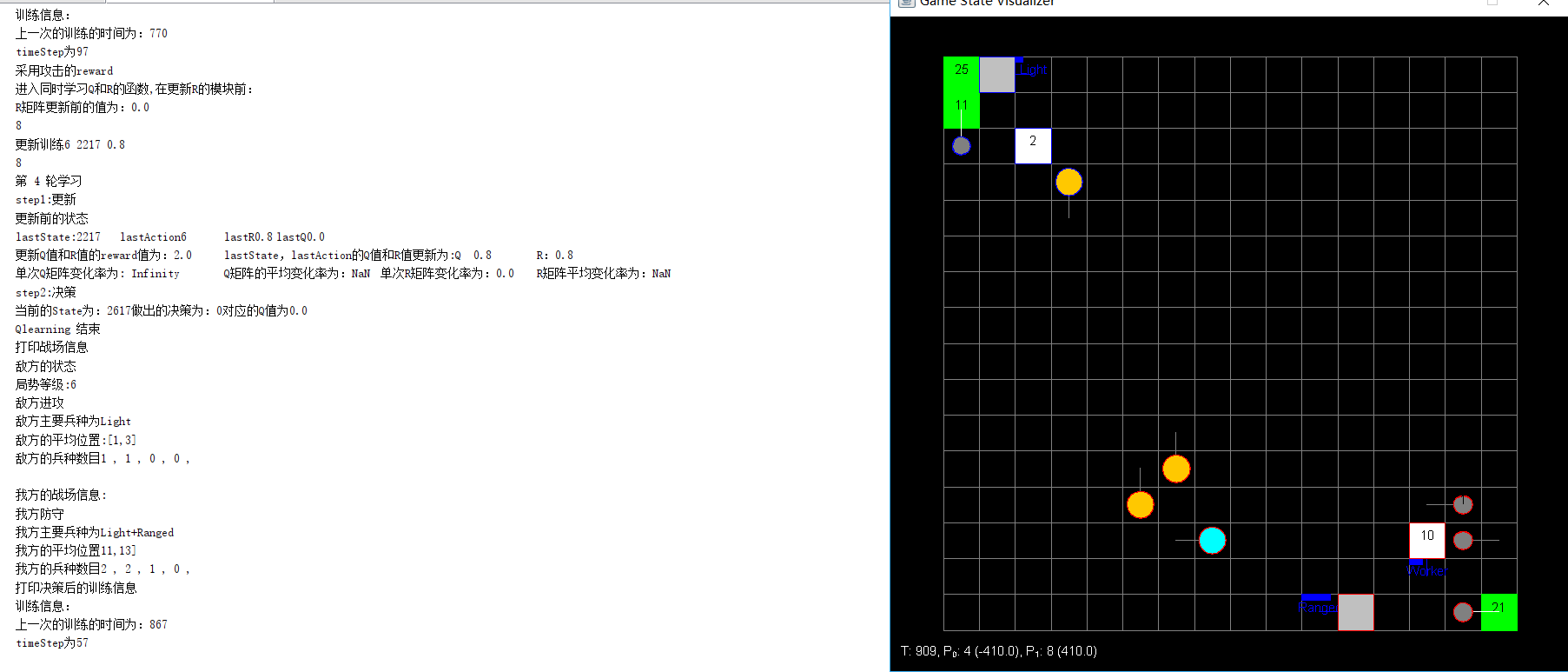


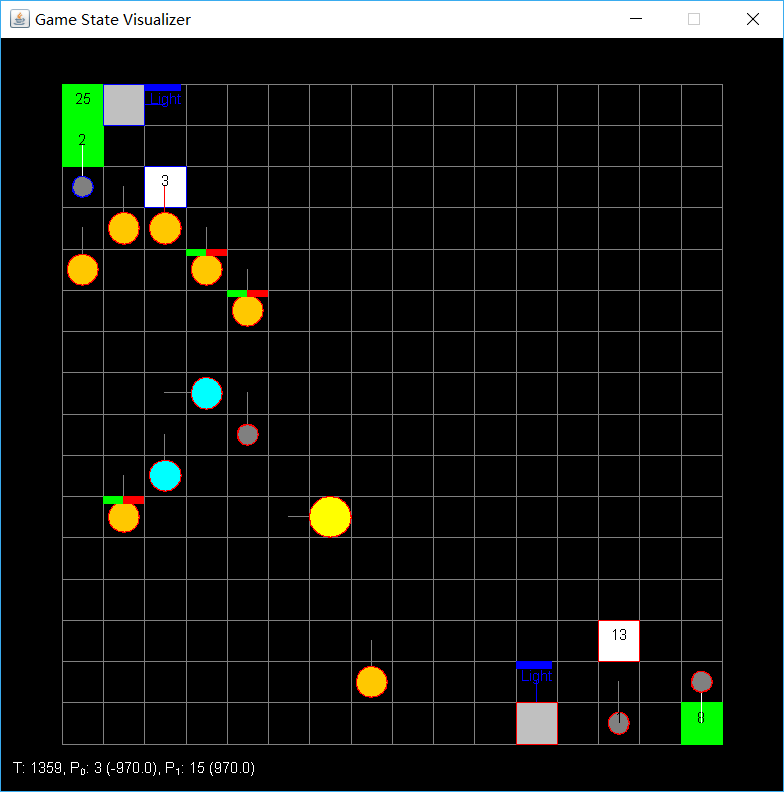
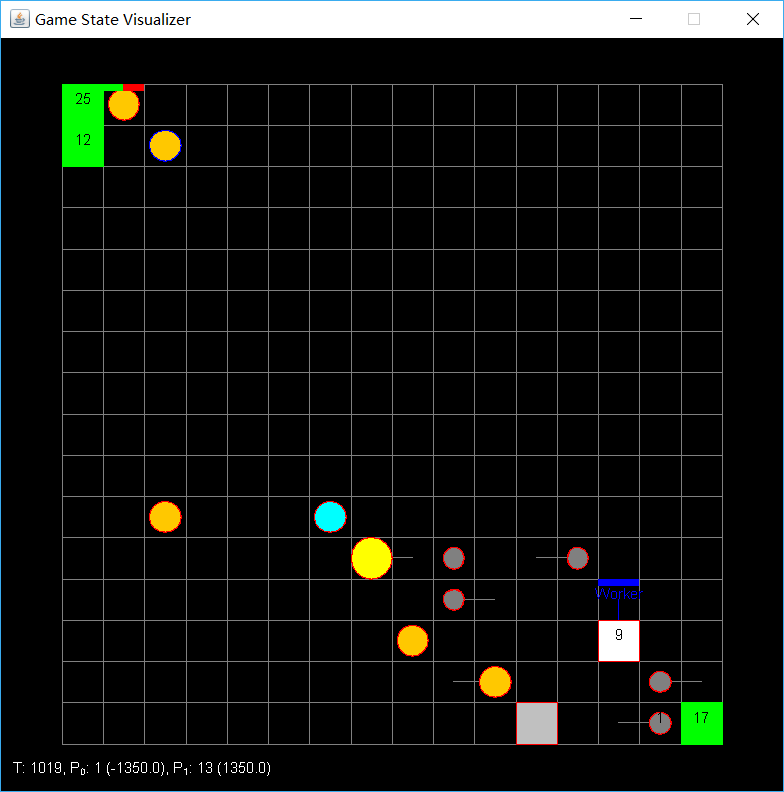
1. 右下 对战CRush\_V1



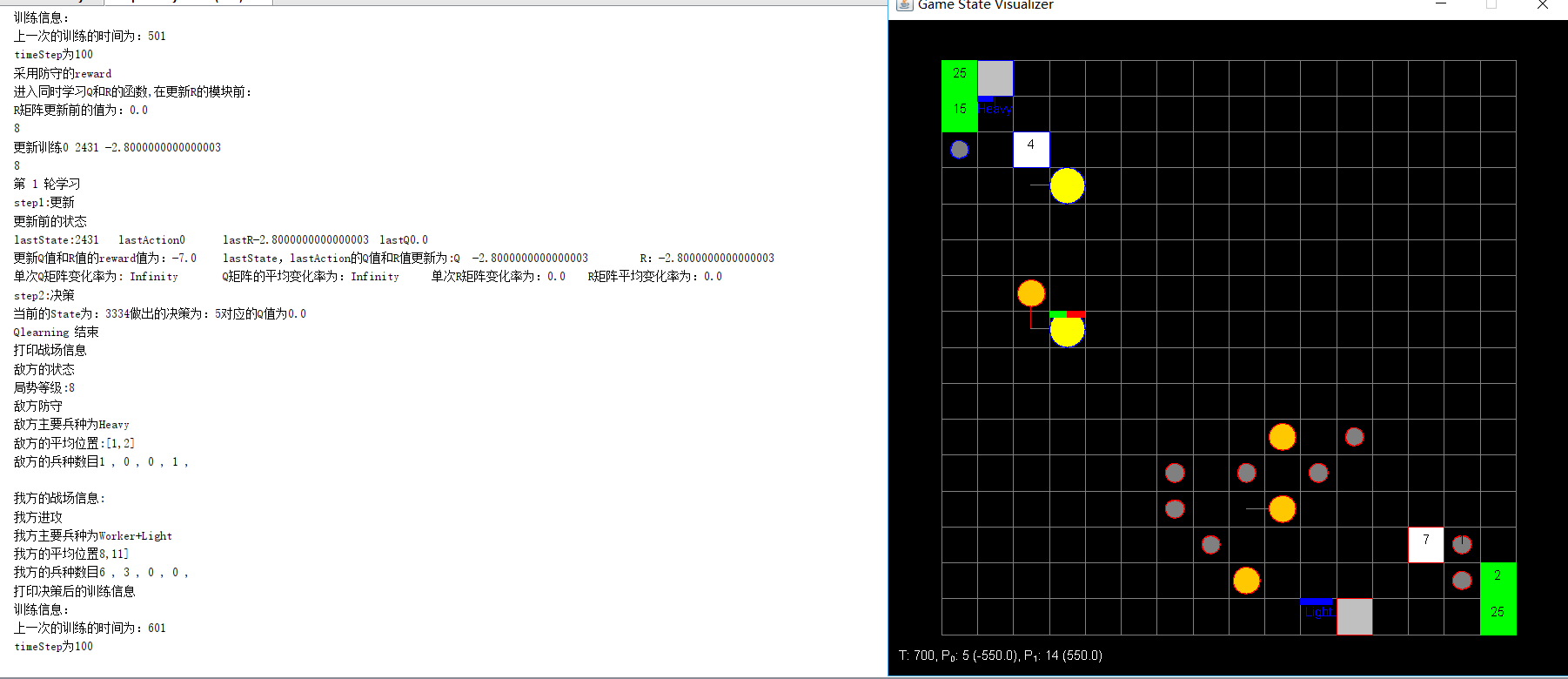


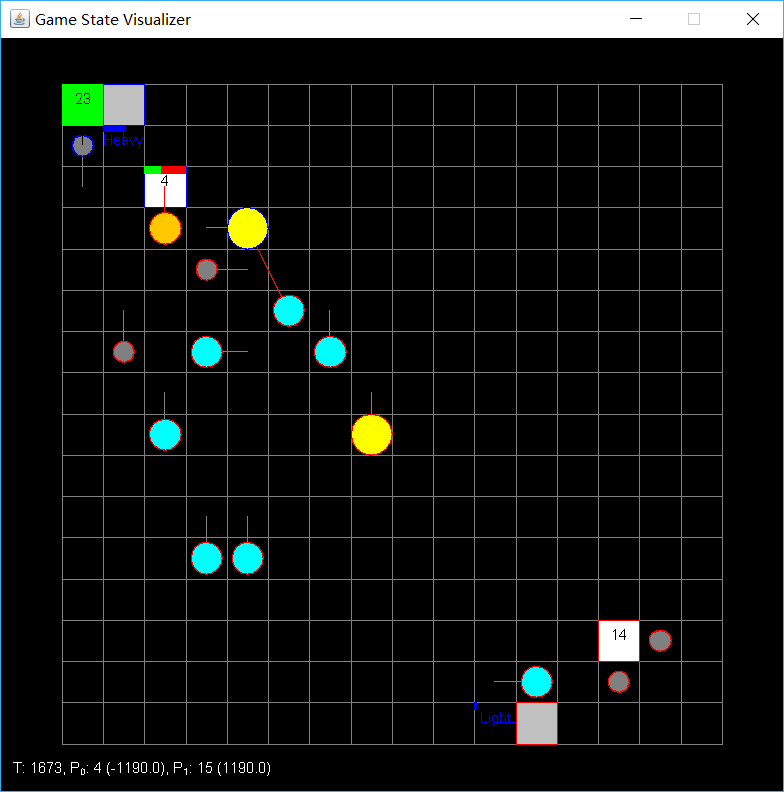
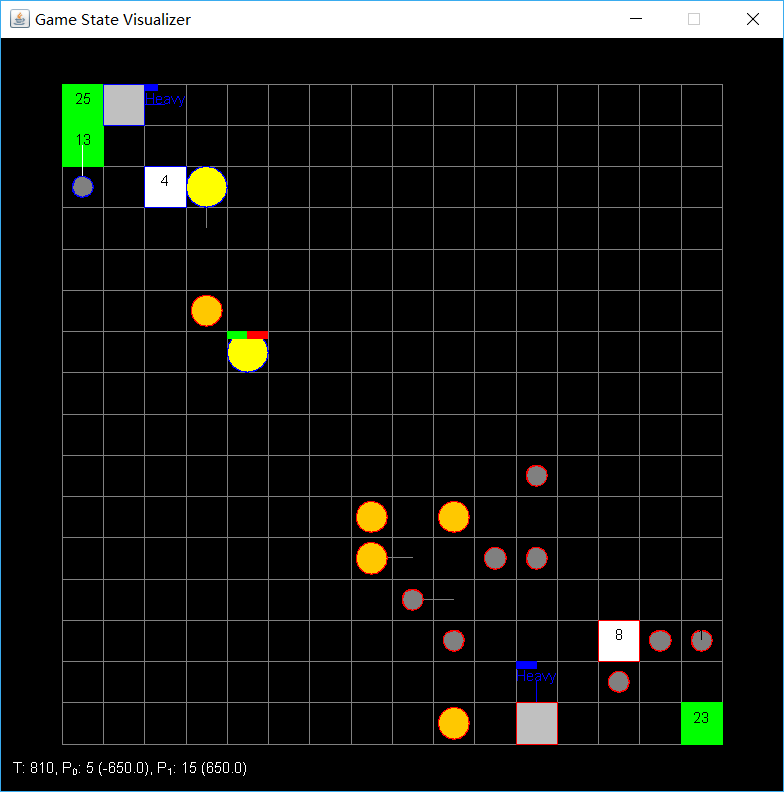
1. 右下 对战LightRush



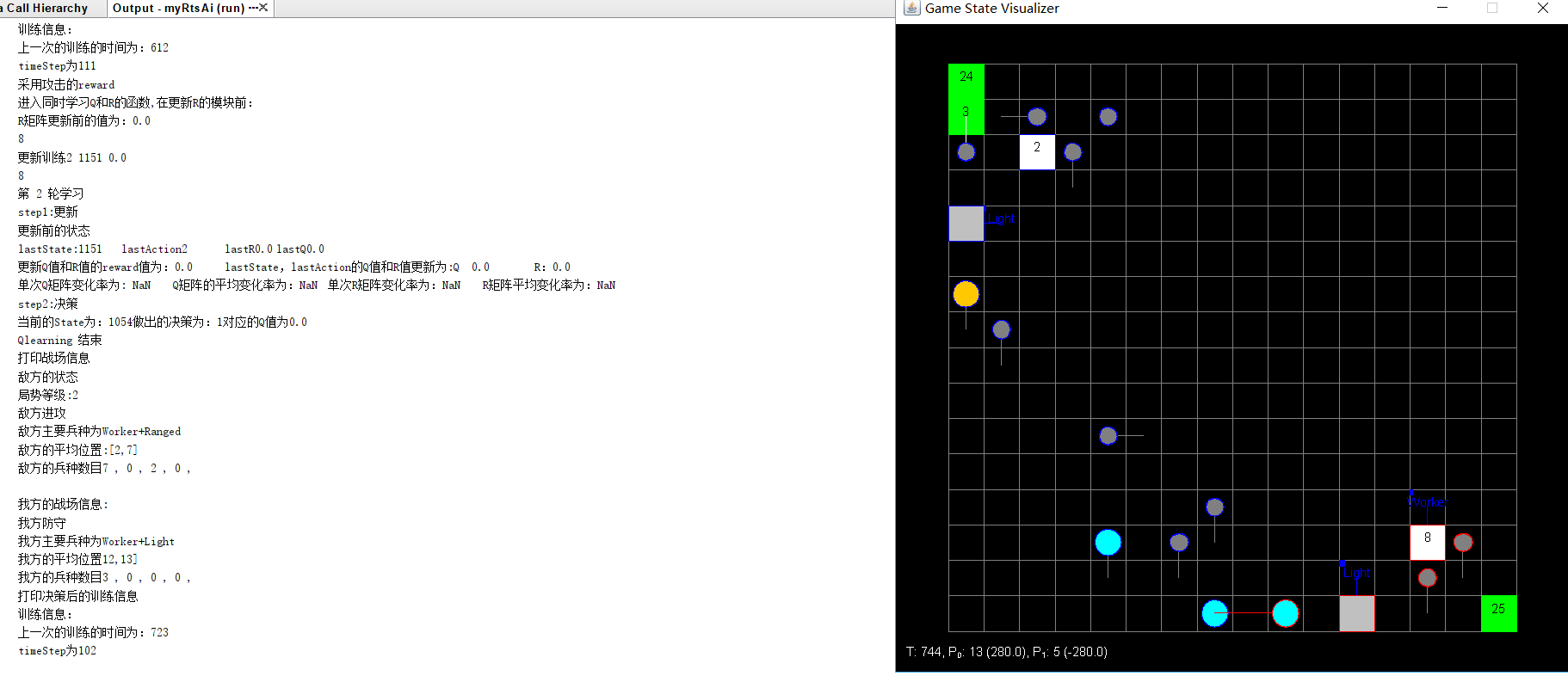


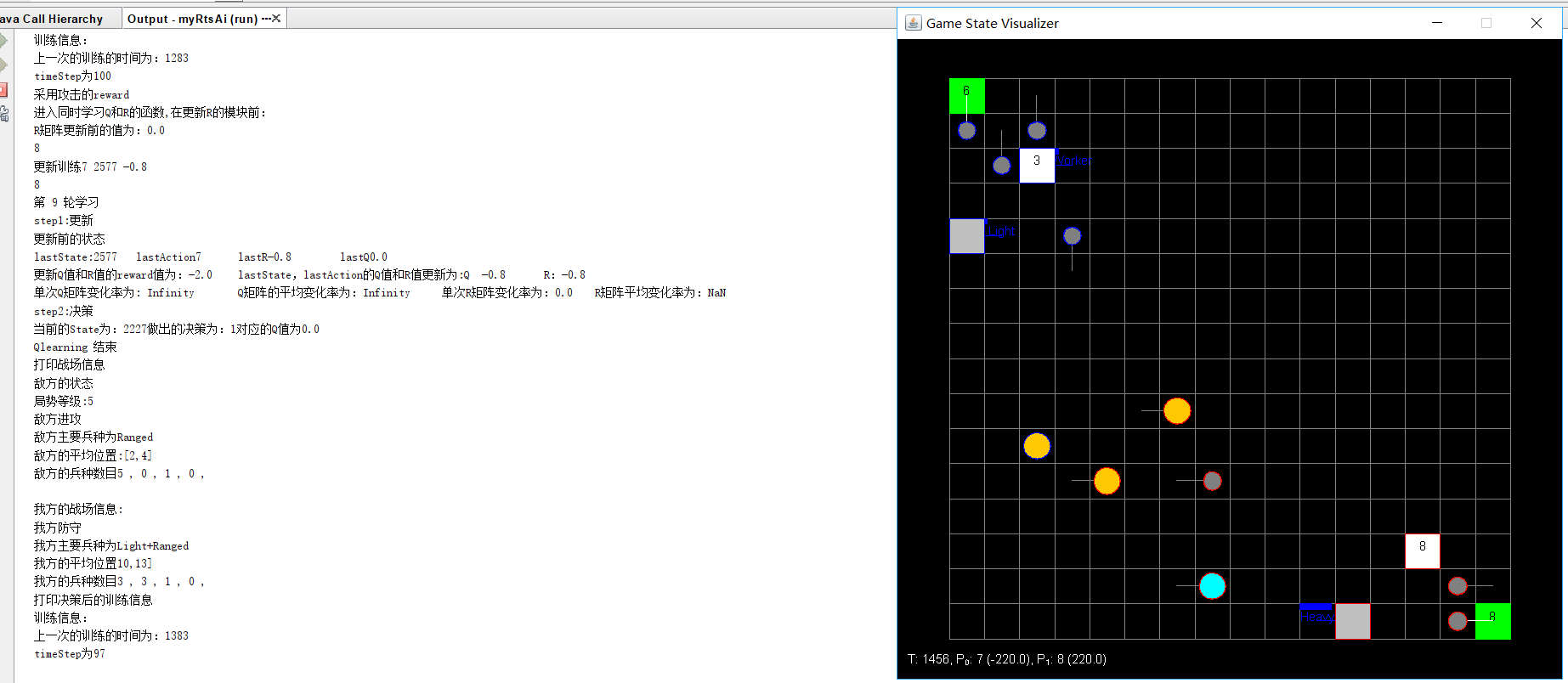
1. 右下 对战HeavyRush

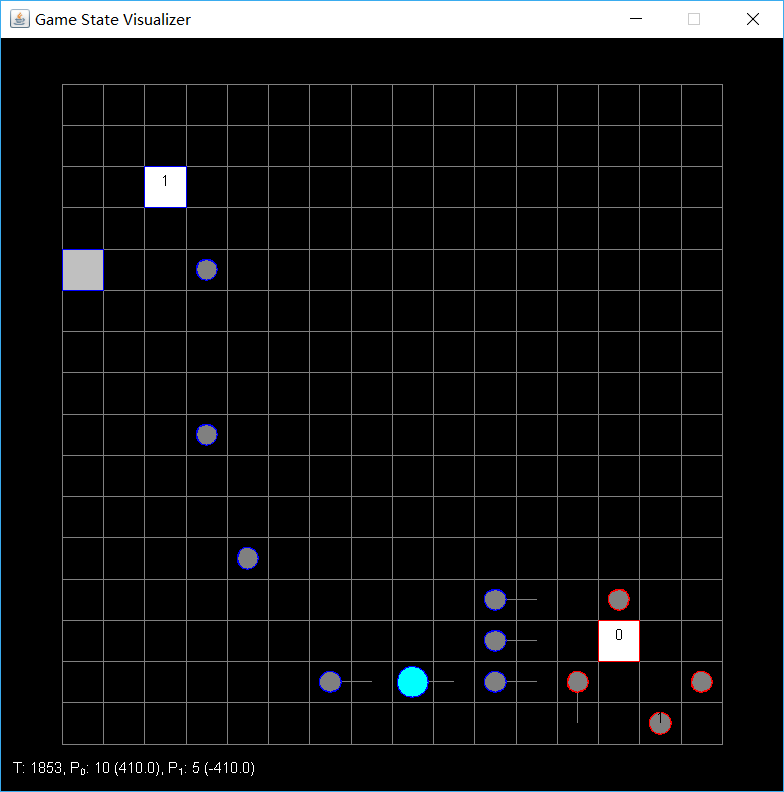
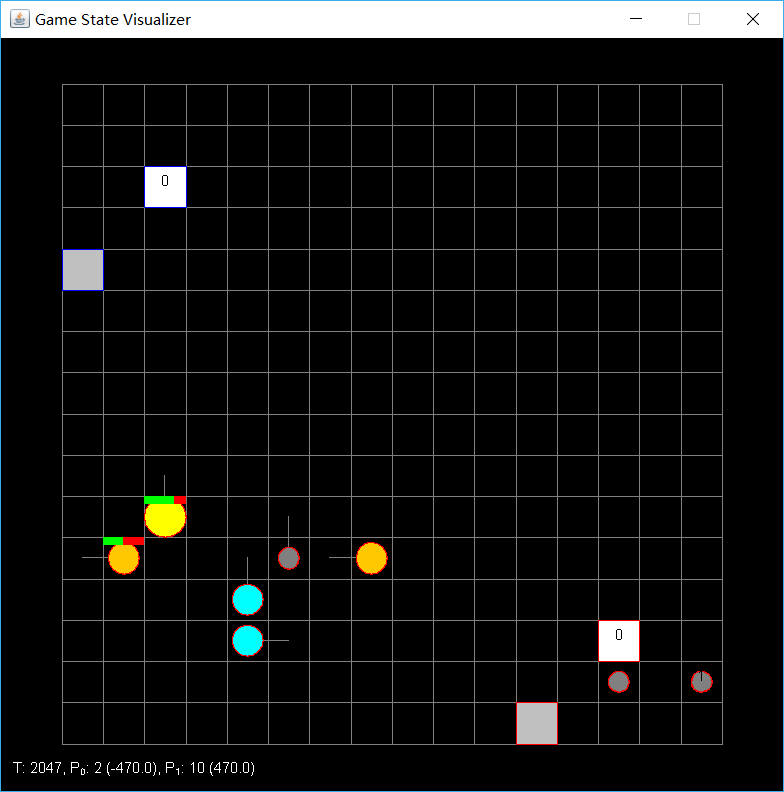




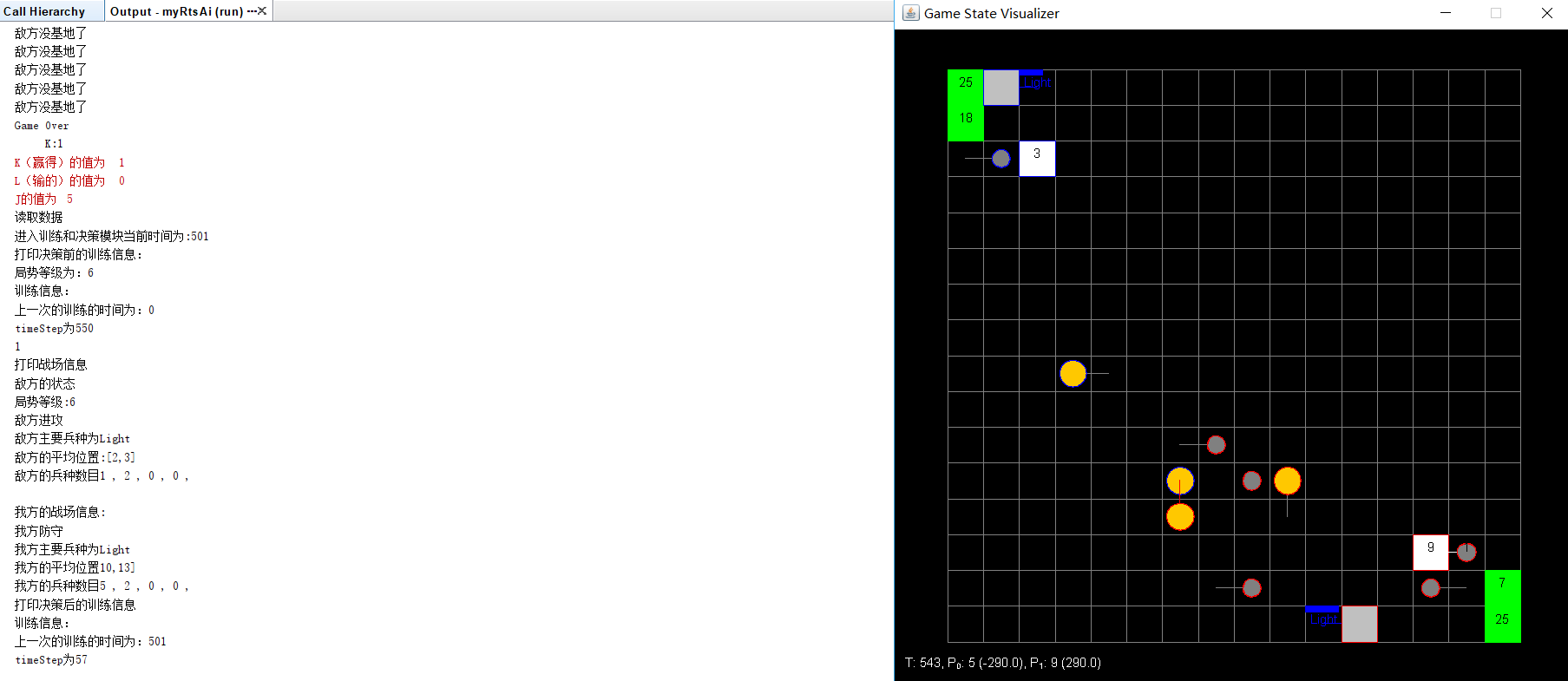
1. 右下 对战同学AI 1号FirstRush

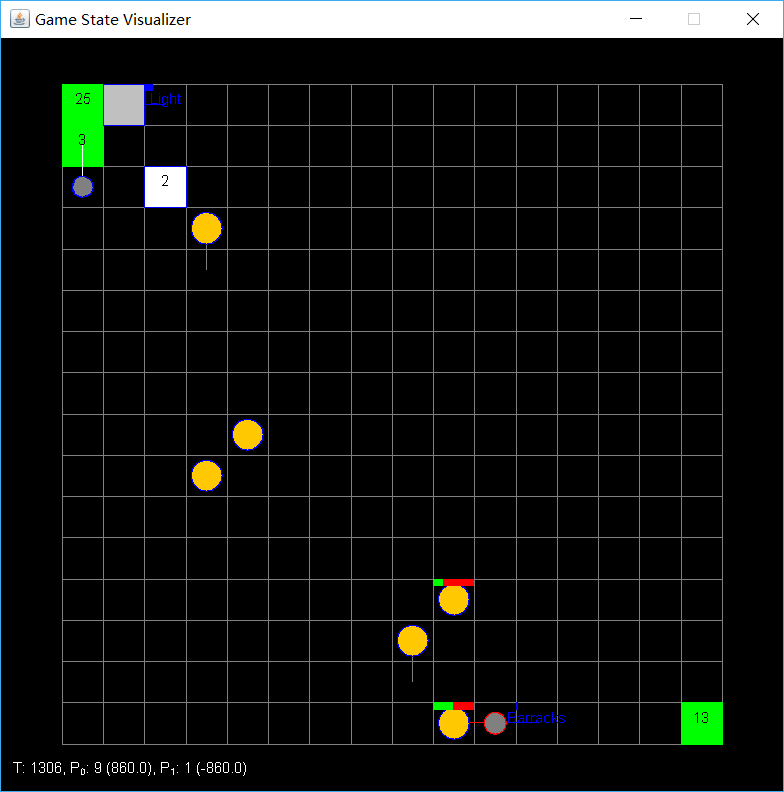
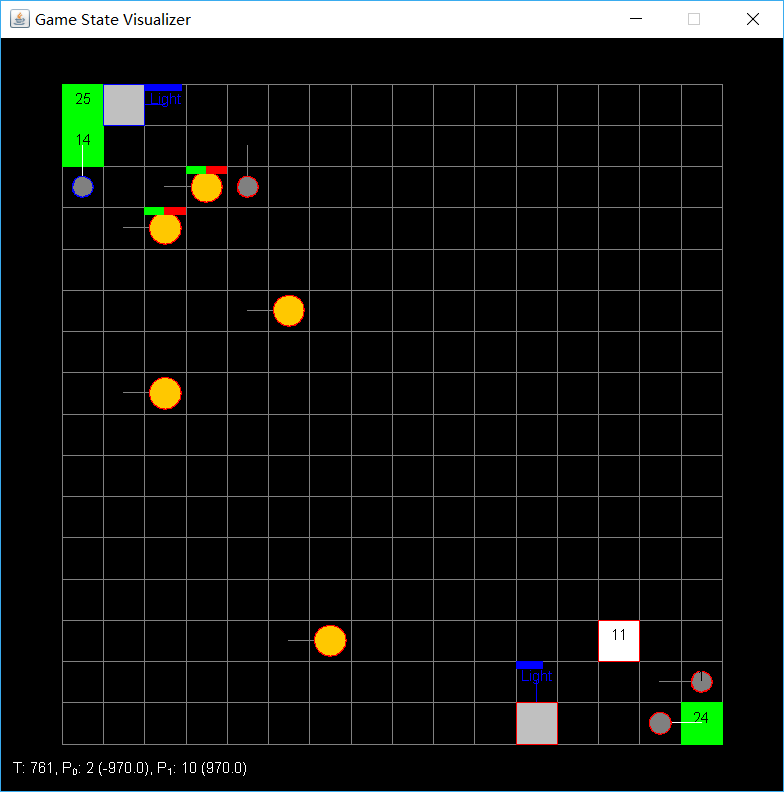




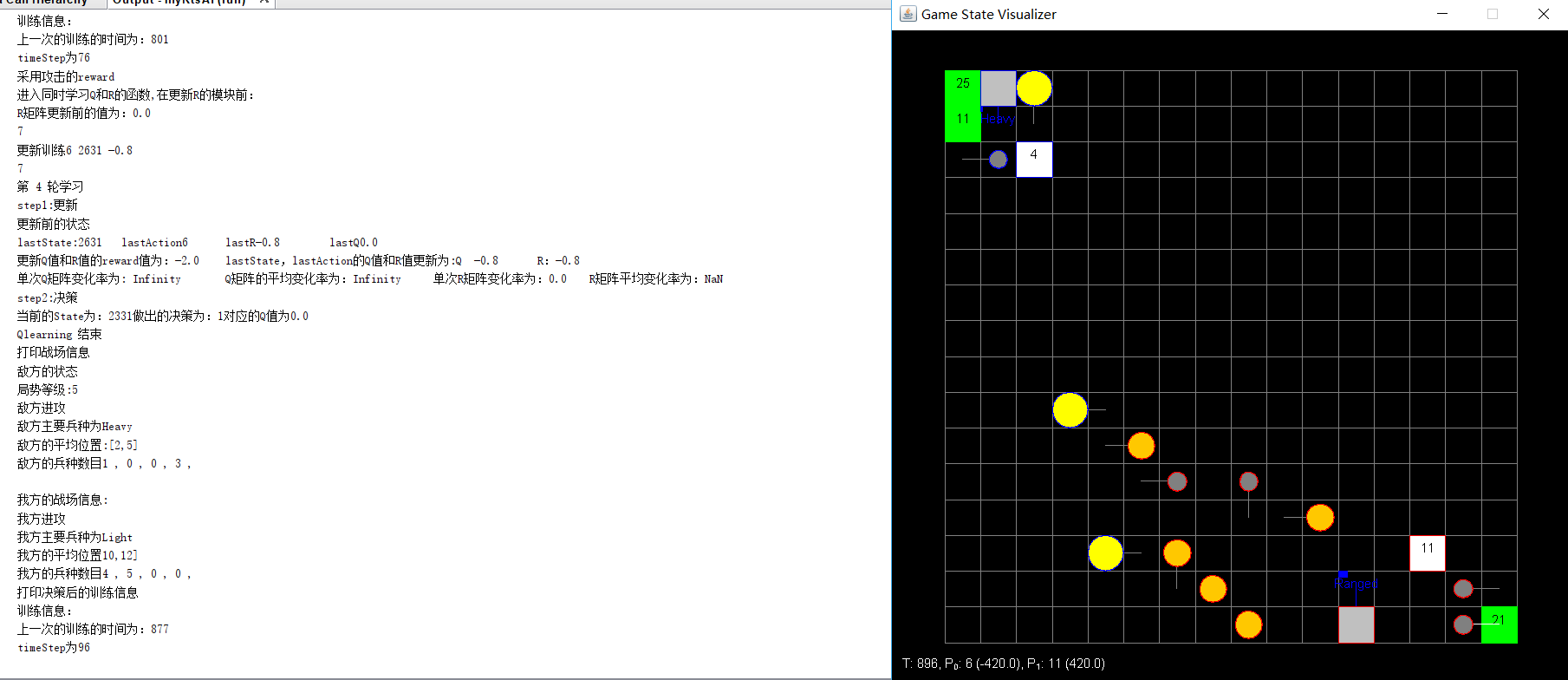


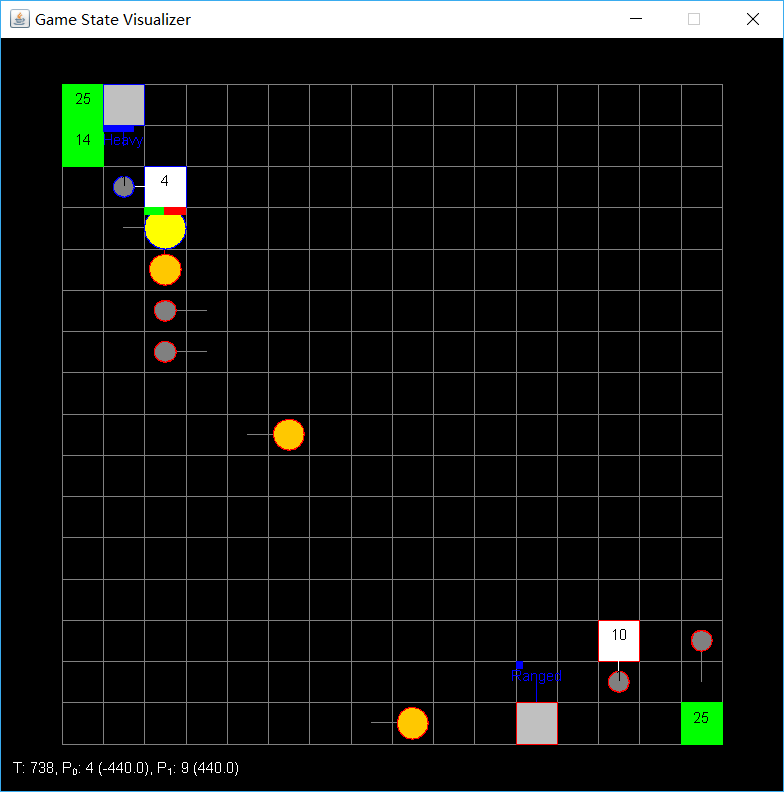
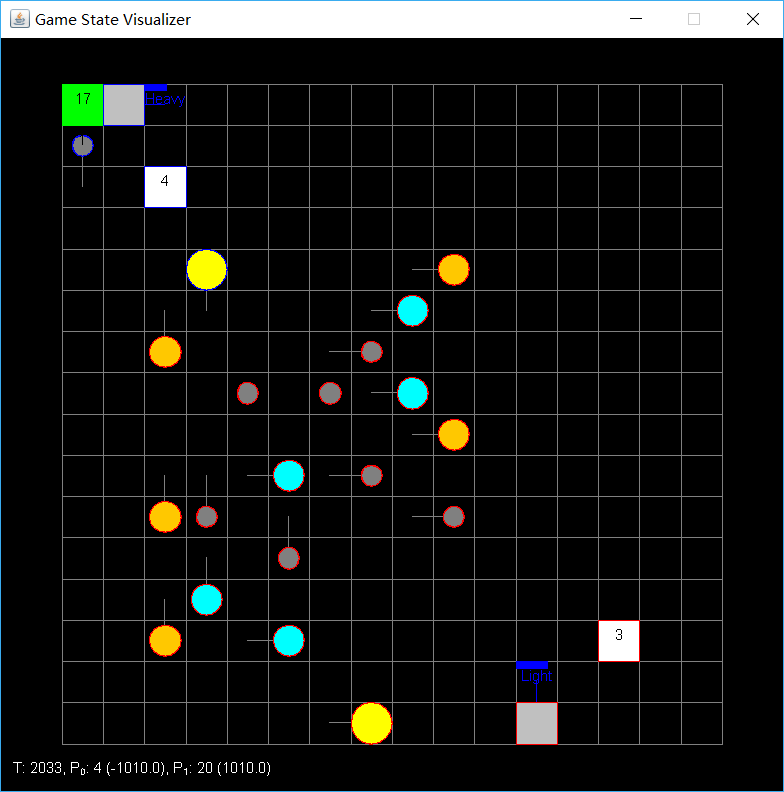
1. 右下 对战LightDefense



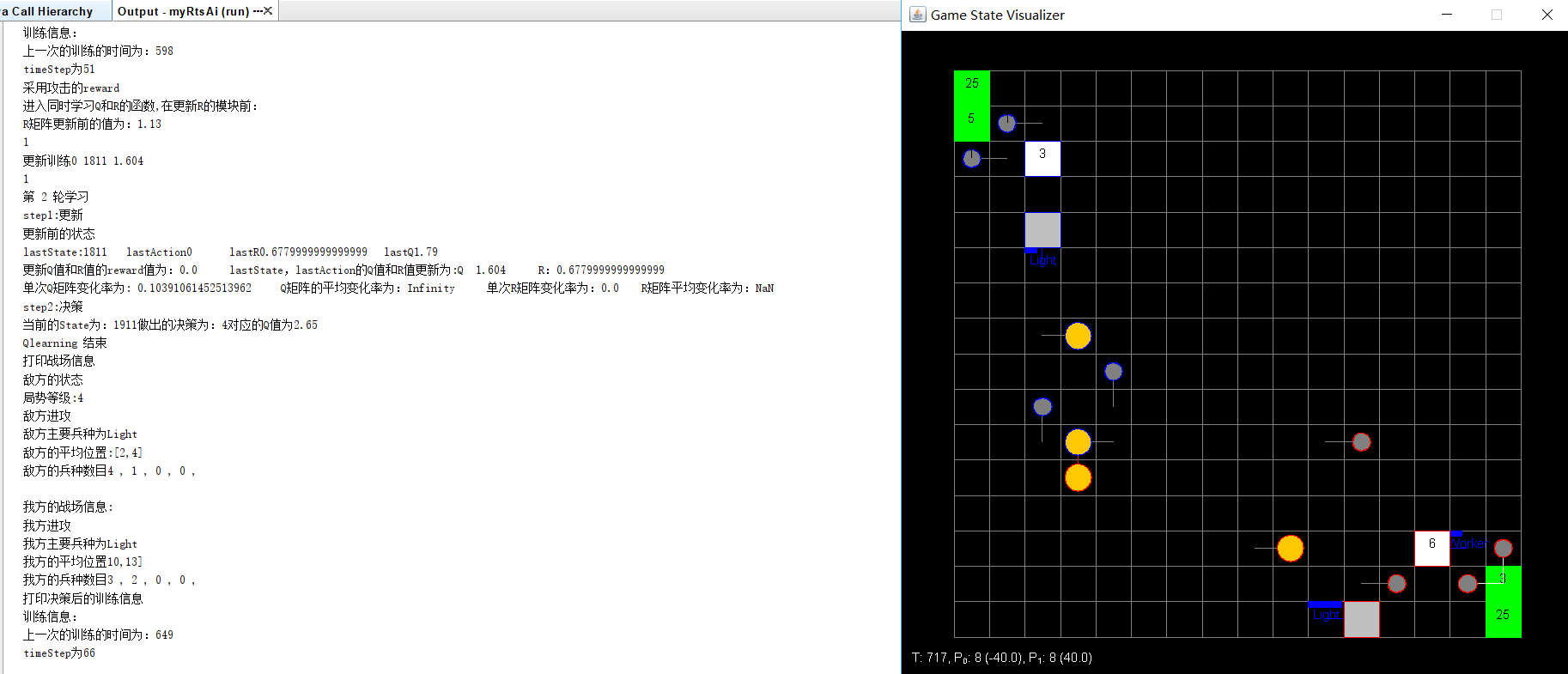


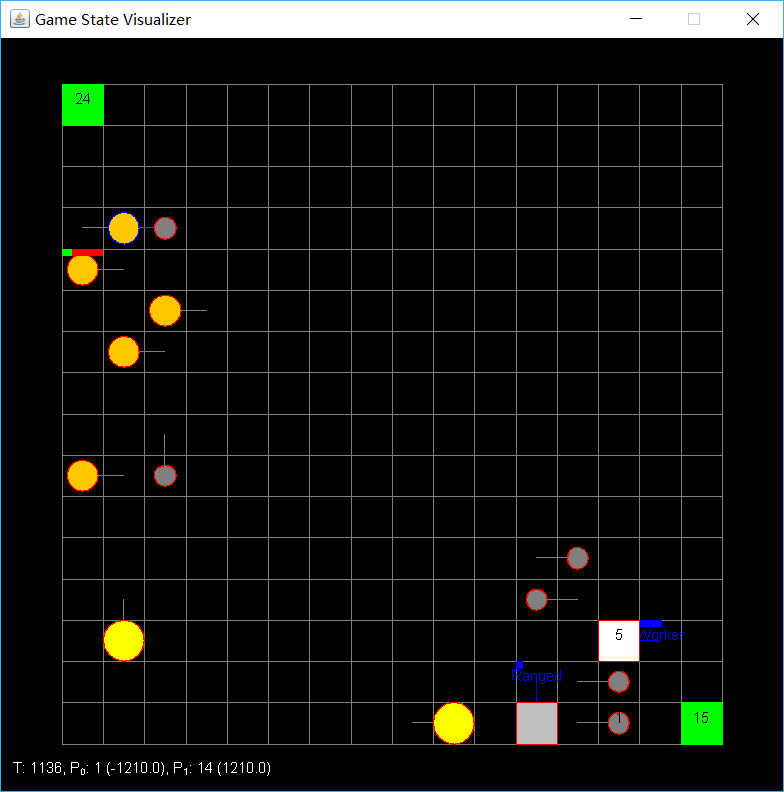
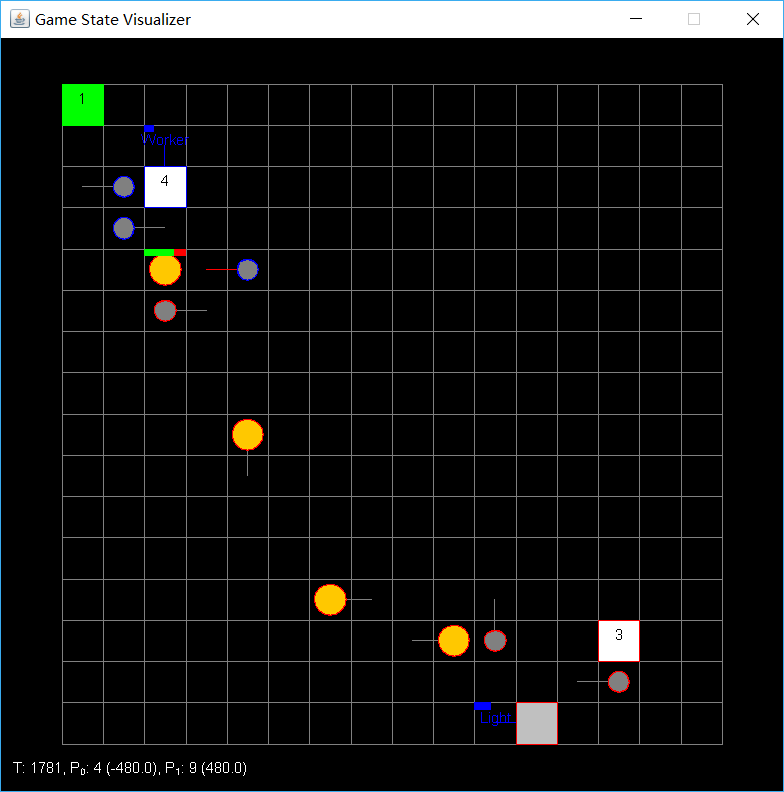
1. 右下 对战HeavyDefense



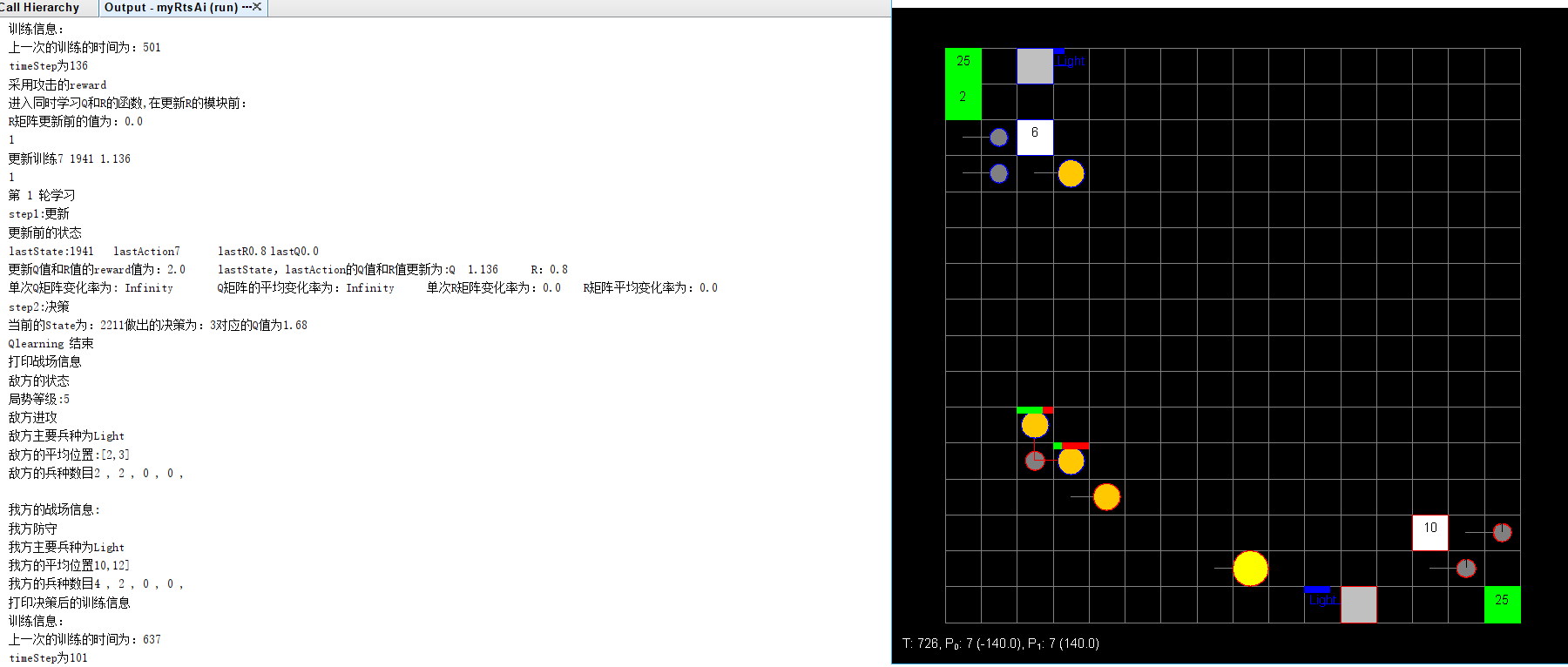


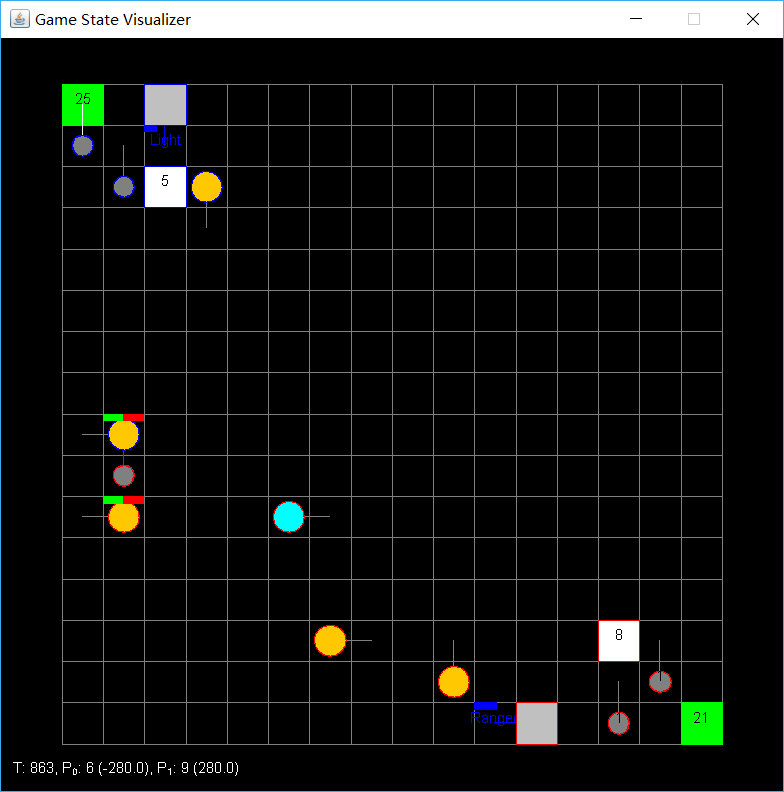
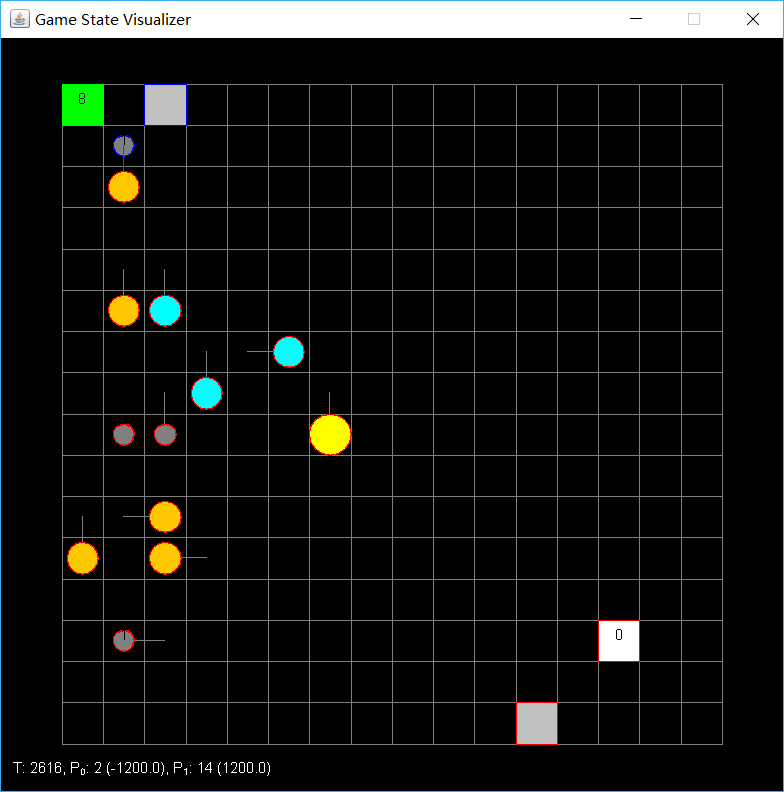
1. 右下 对战同学AI 2号 bottom

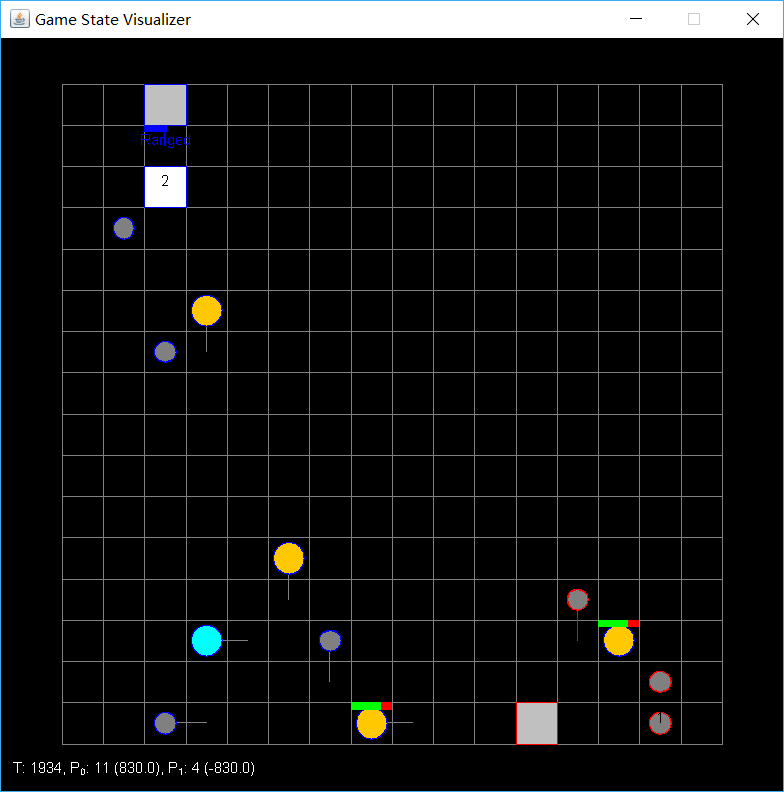
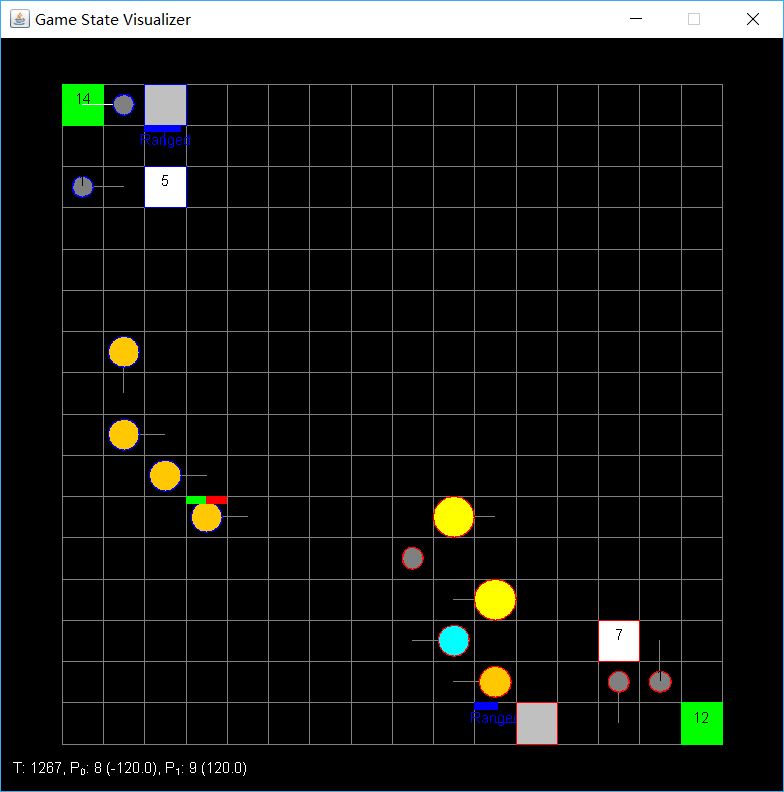




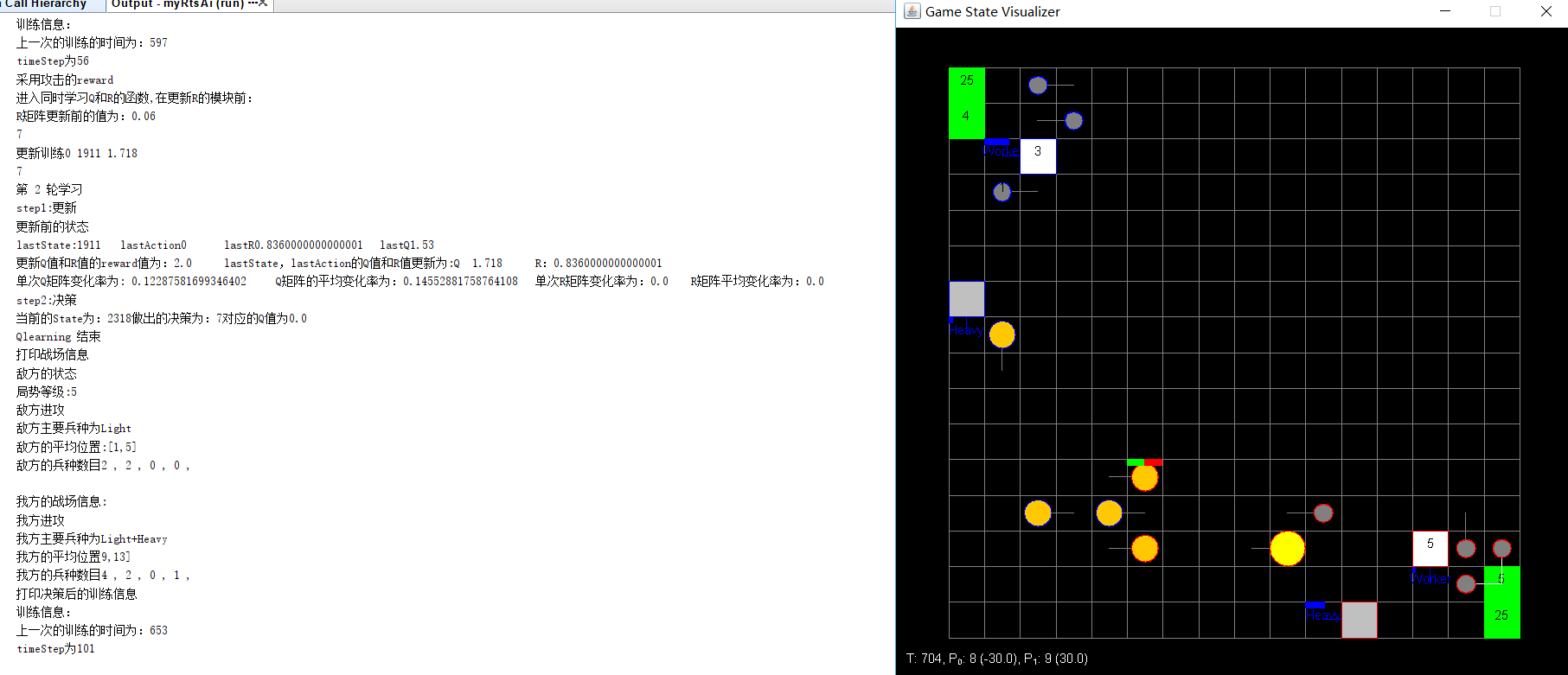
1. 右下 对战同学AI 3号 Simulate

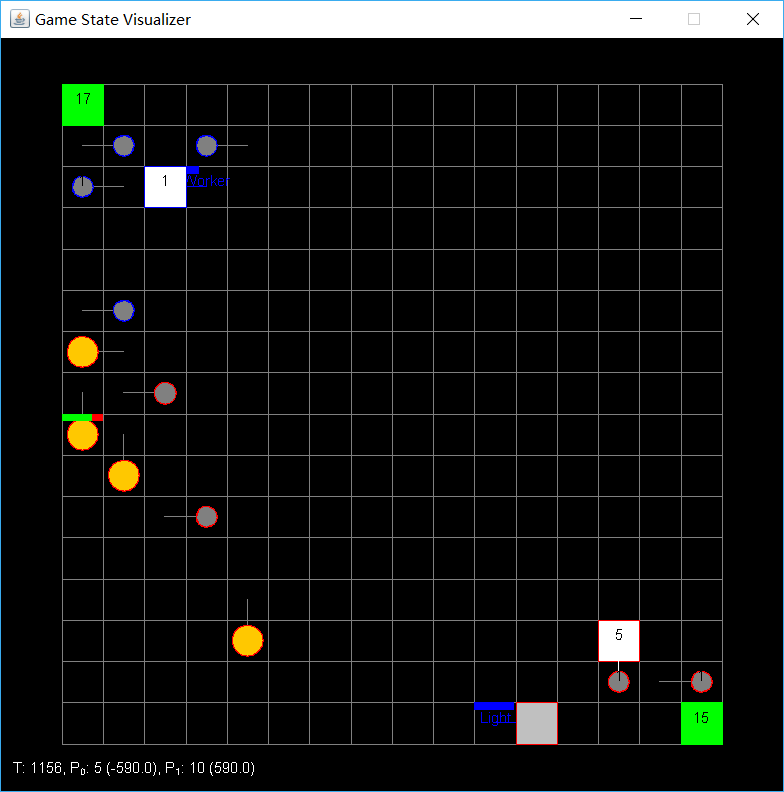
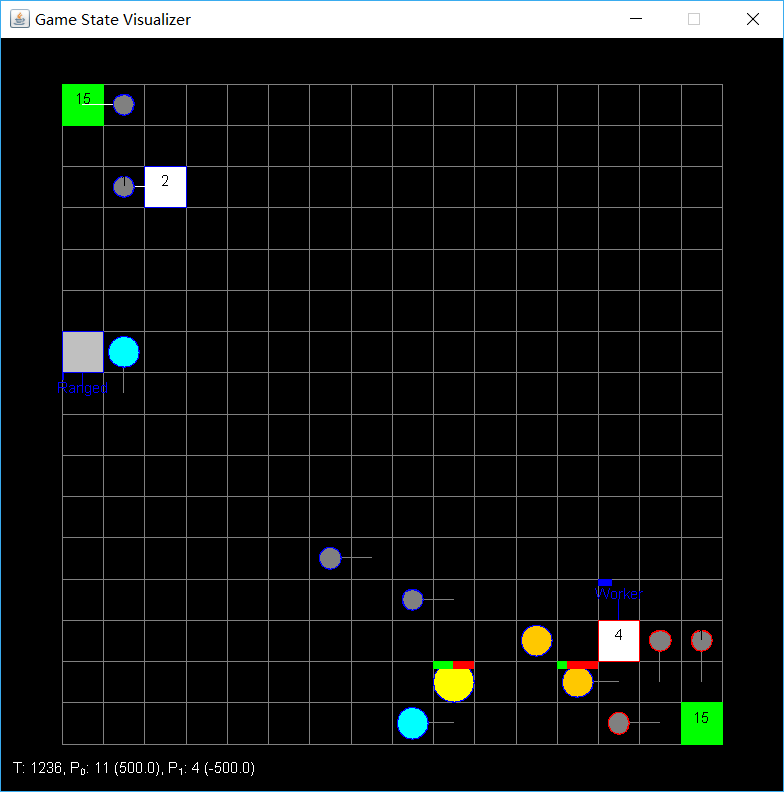




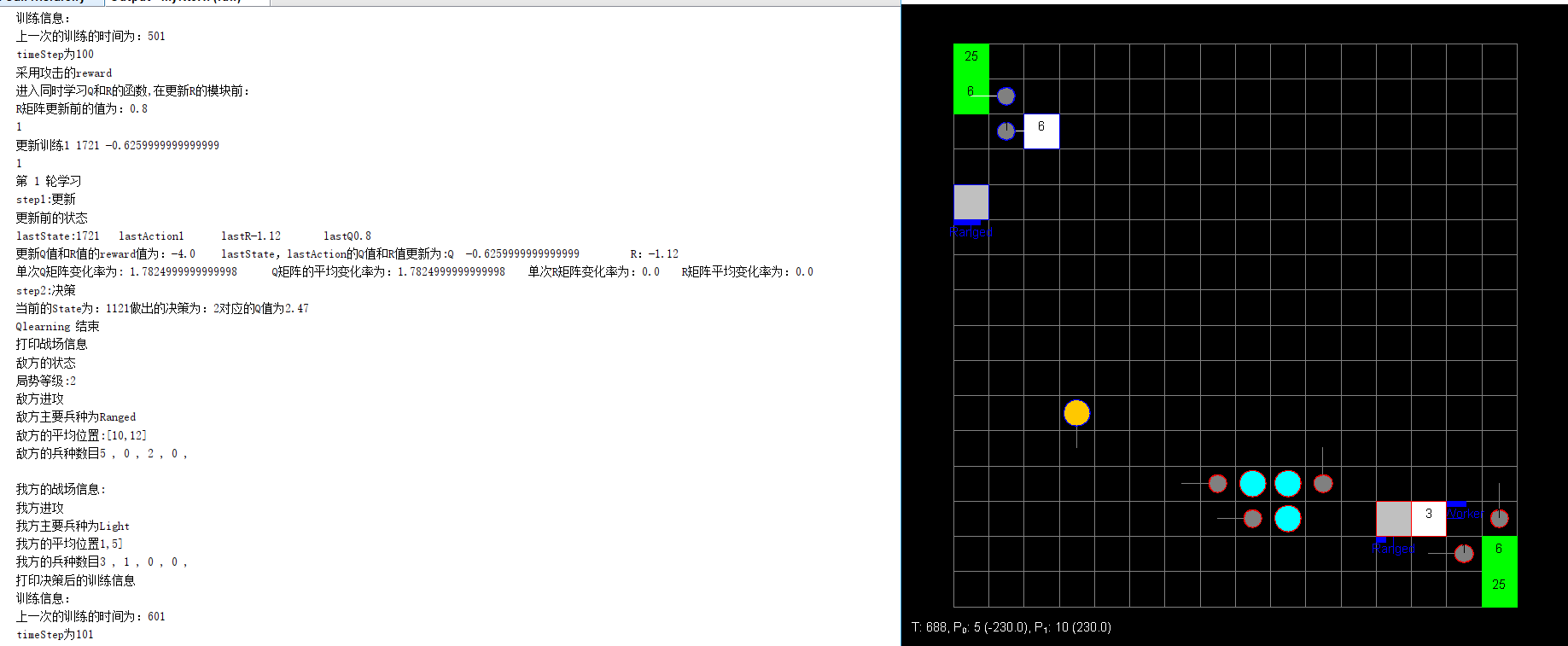


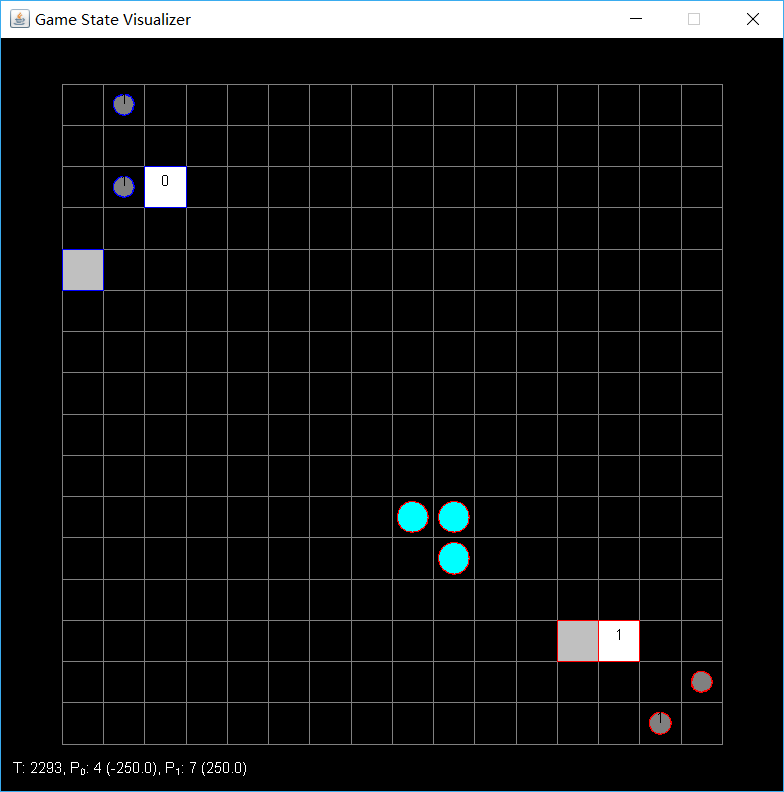
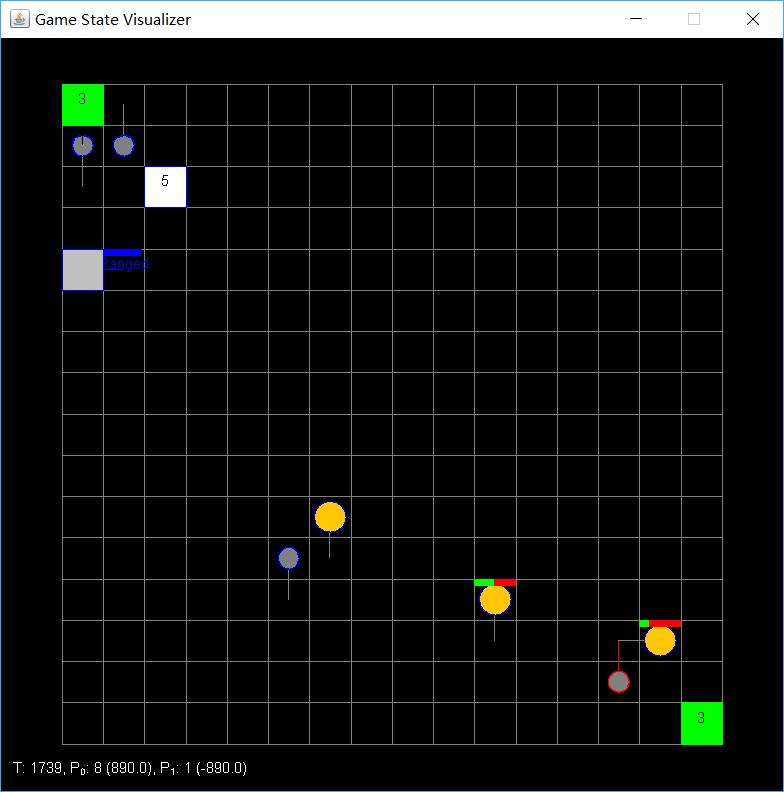
1. 右下 对战同学AI 4号 MyNewAI



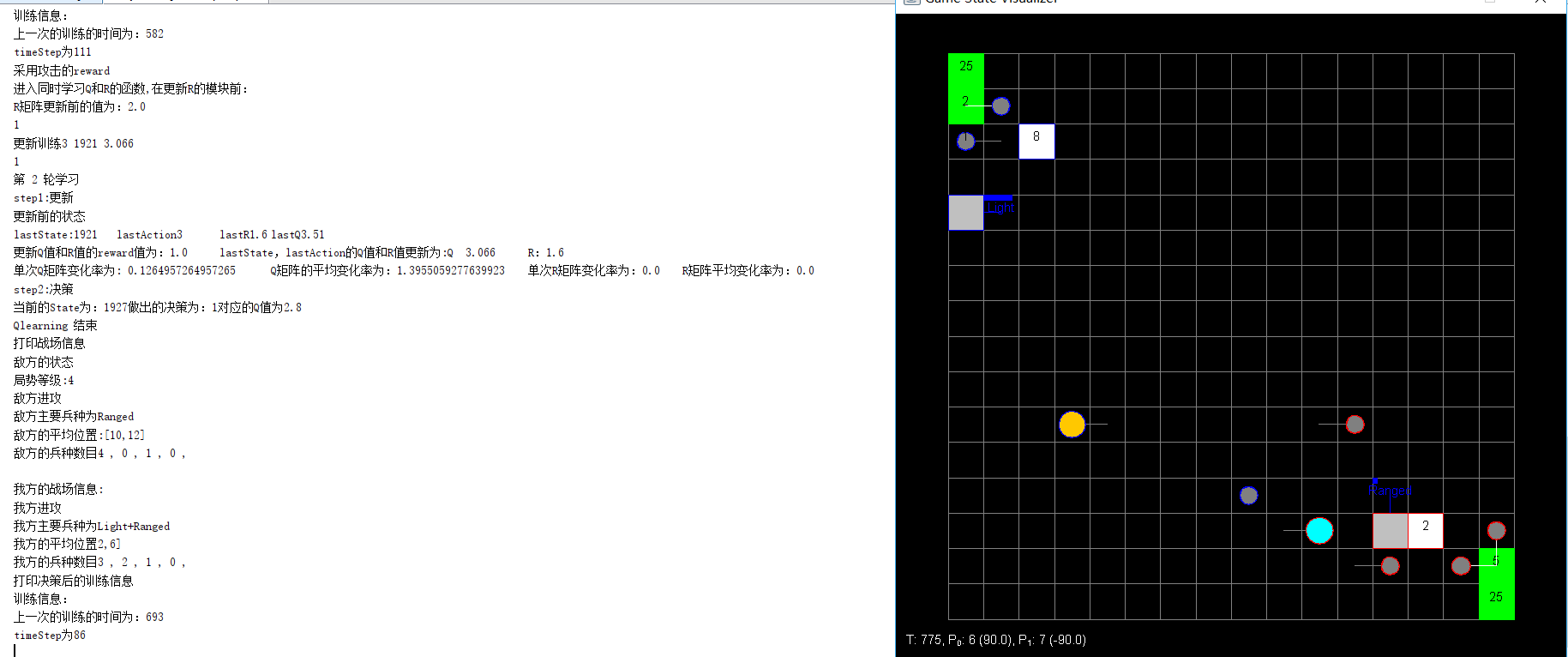


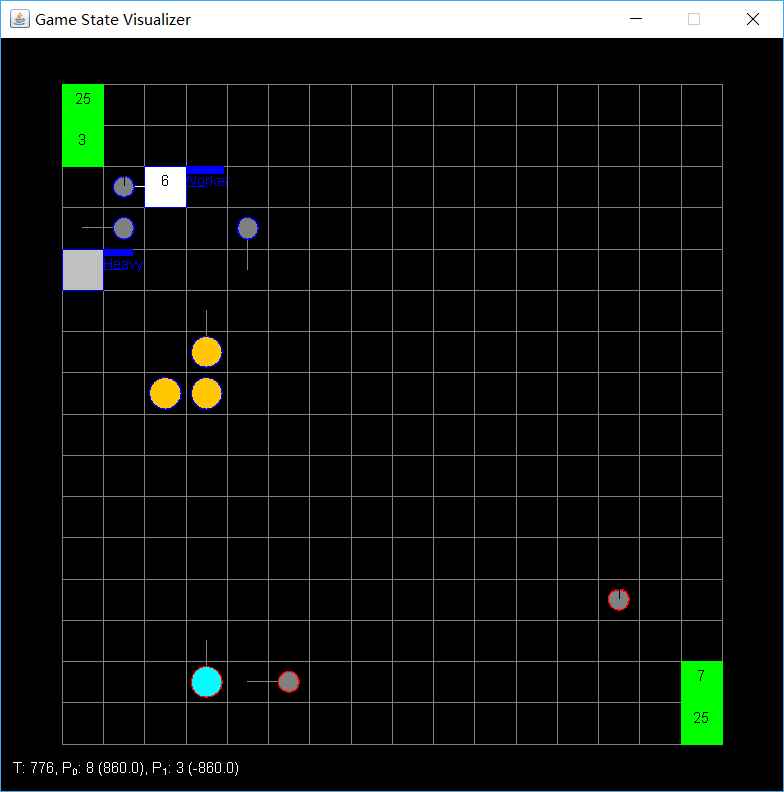
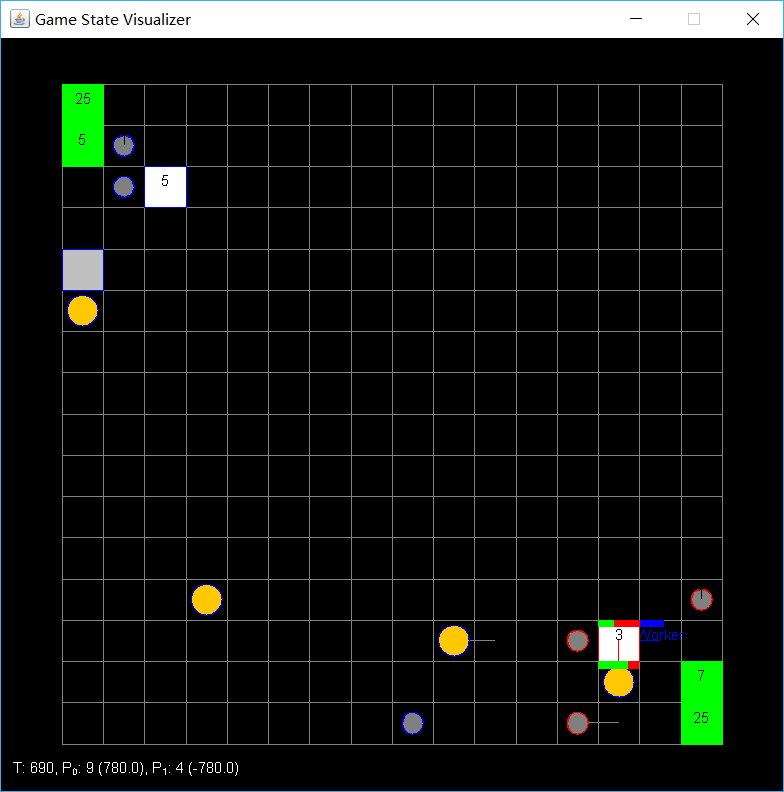
1. 左上 对战CRush\_V2



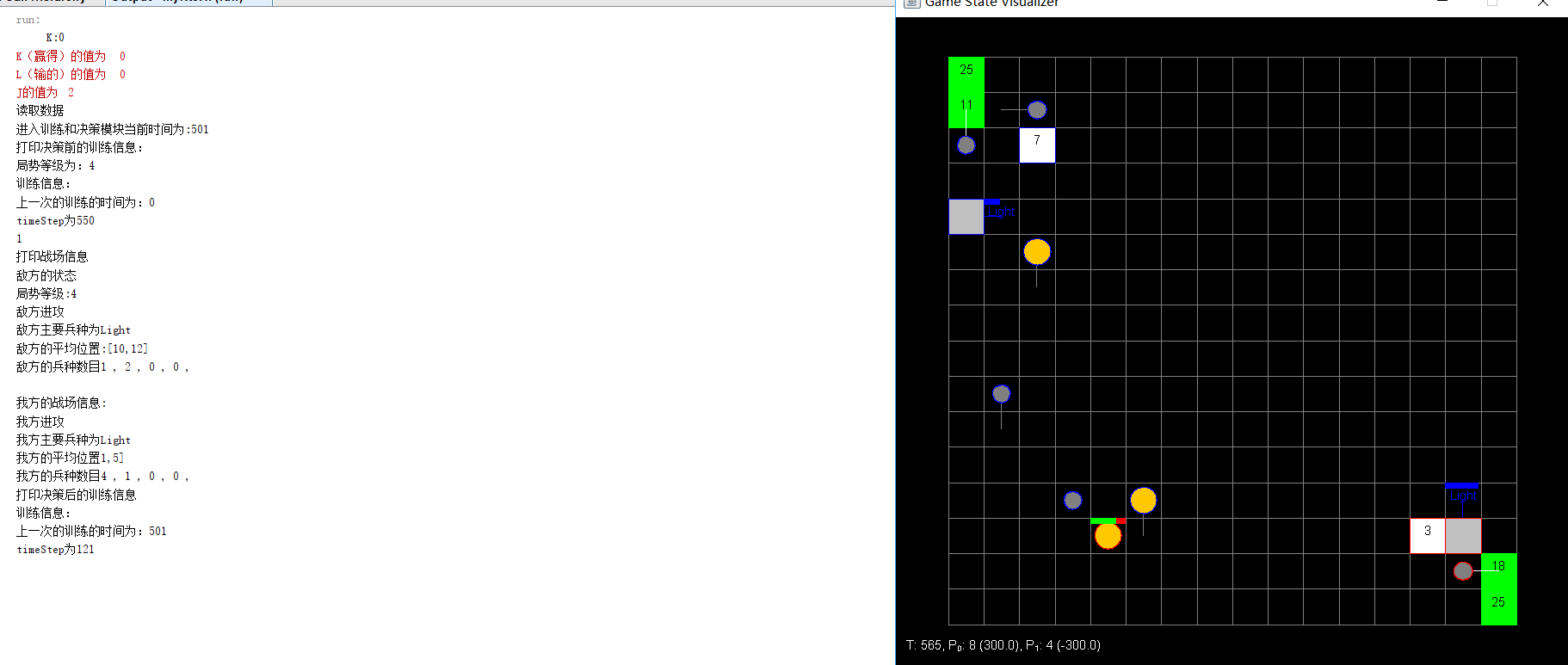


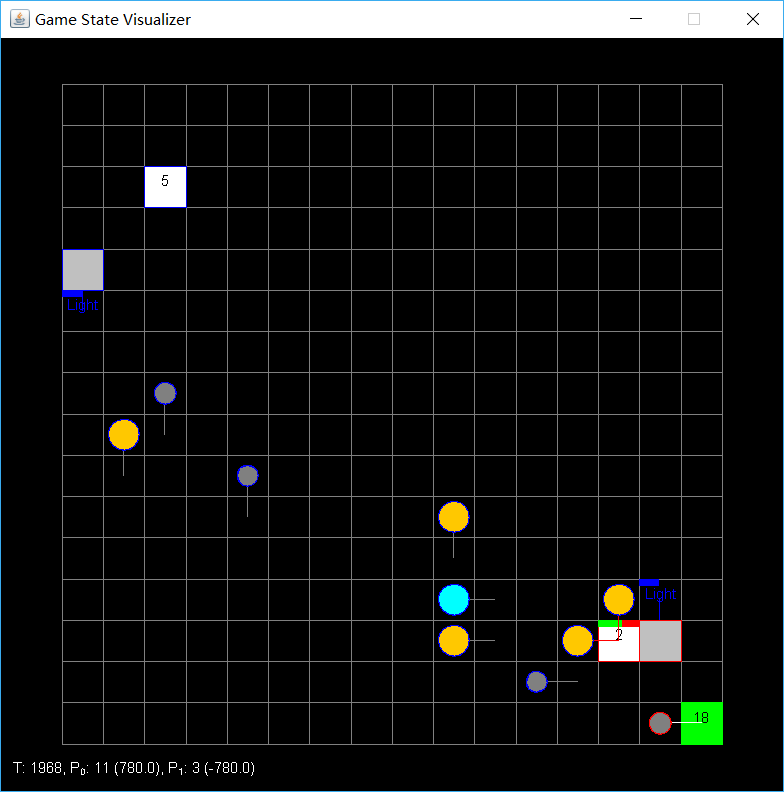
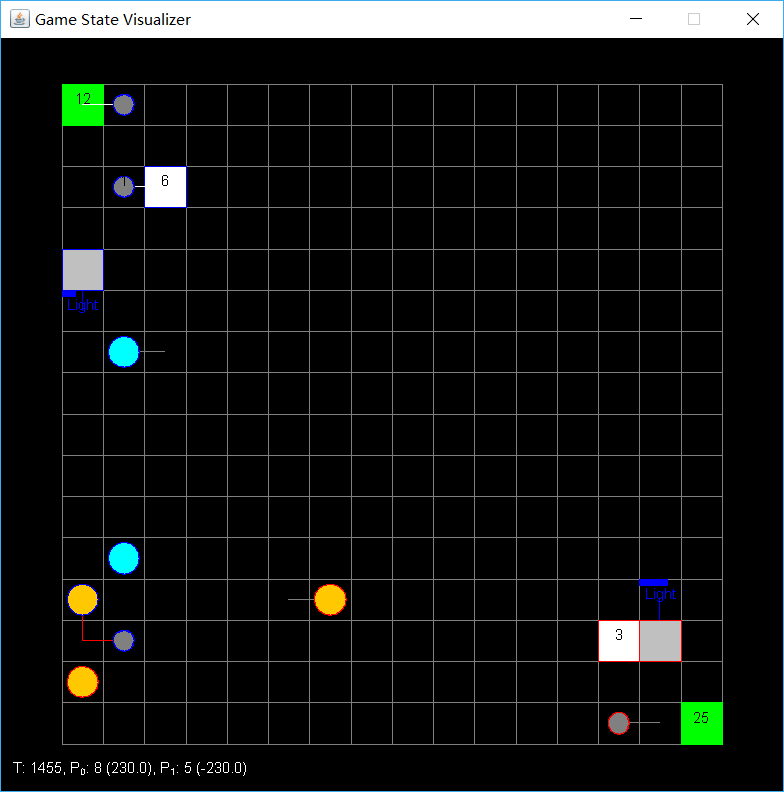
1. 左上 对战CRush\_V1

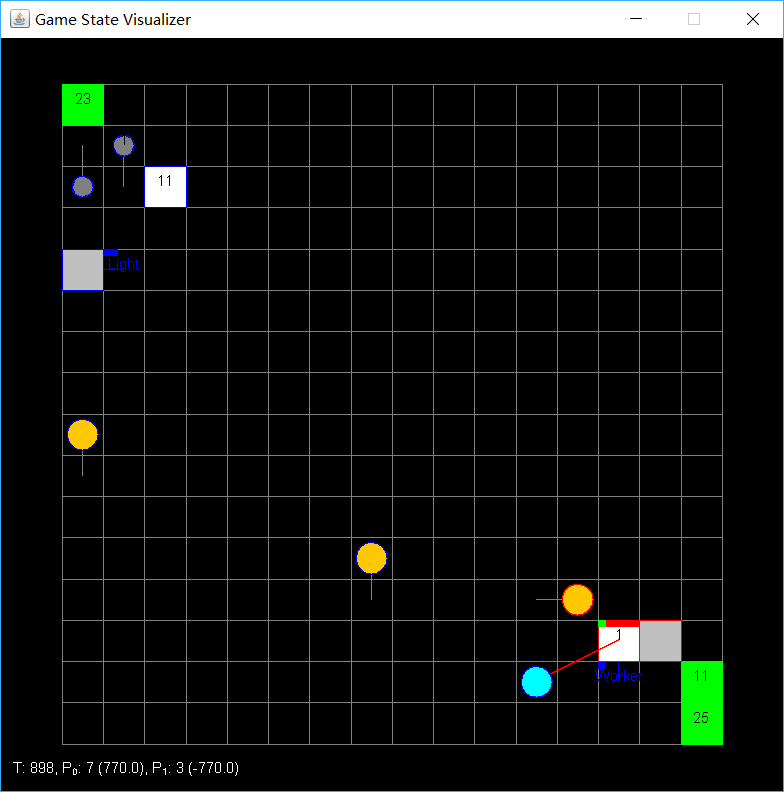
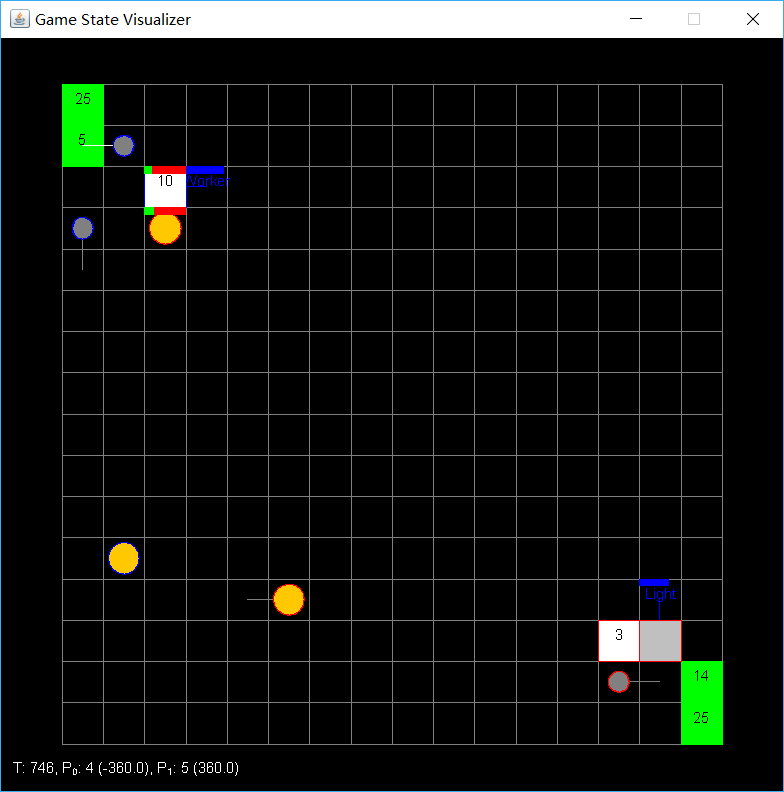




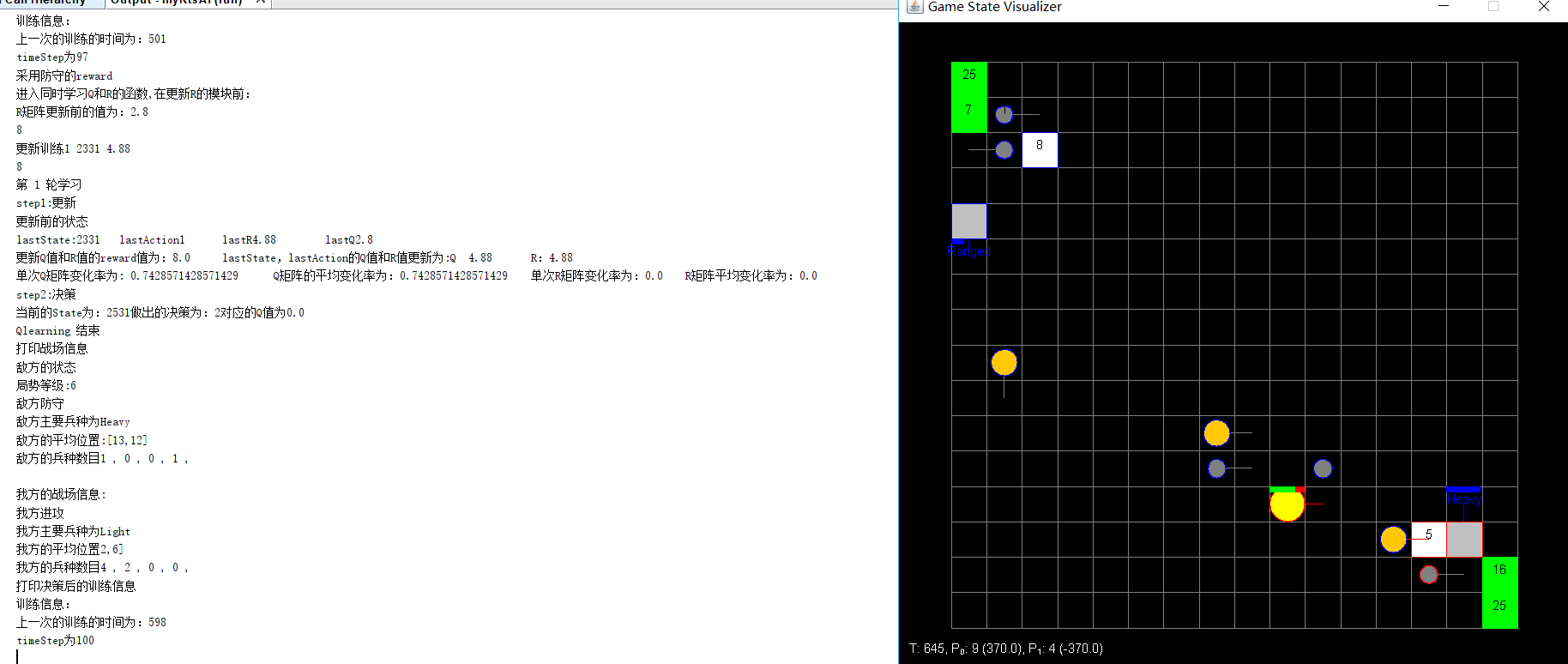
1. 左上 对战LightRush



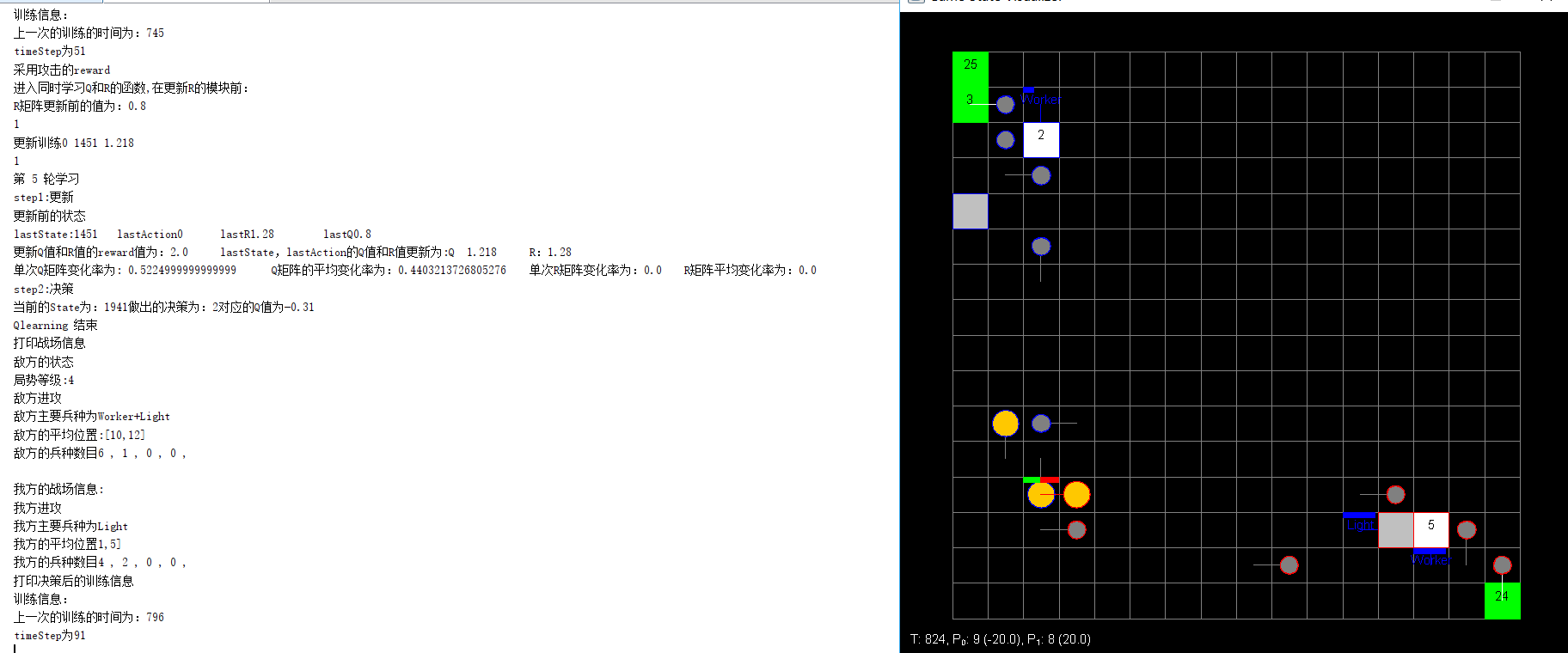


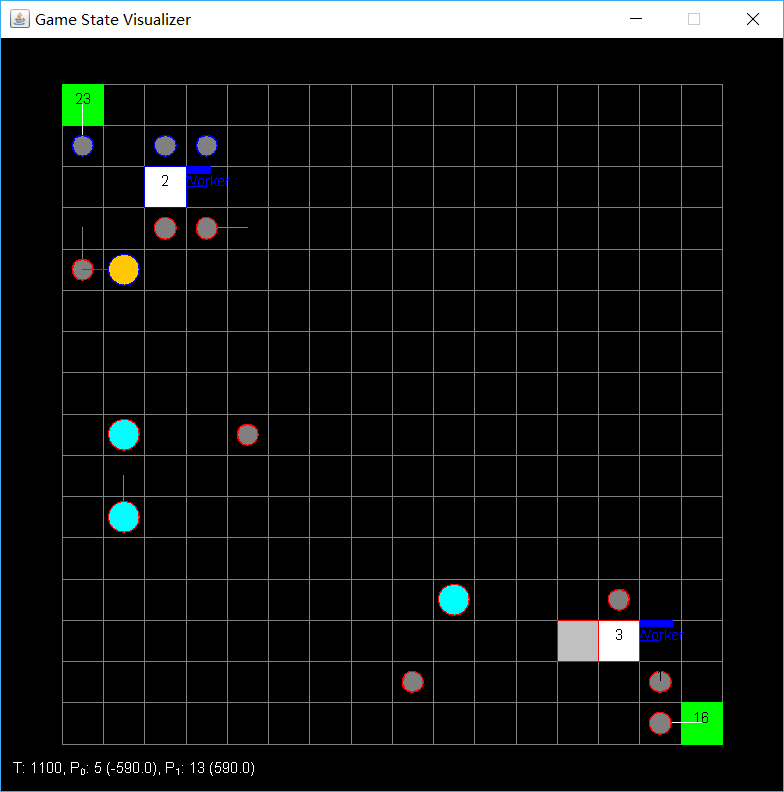
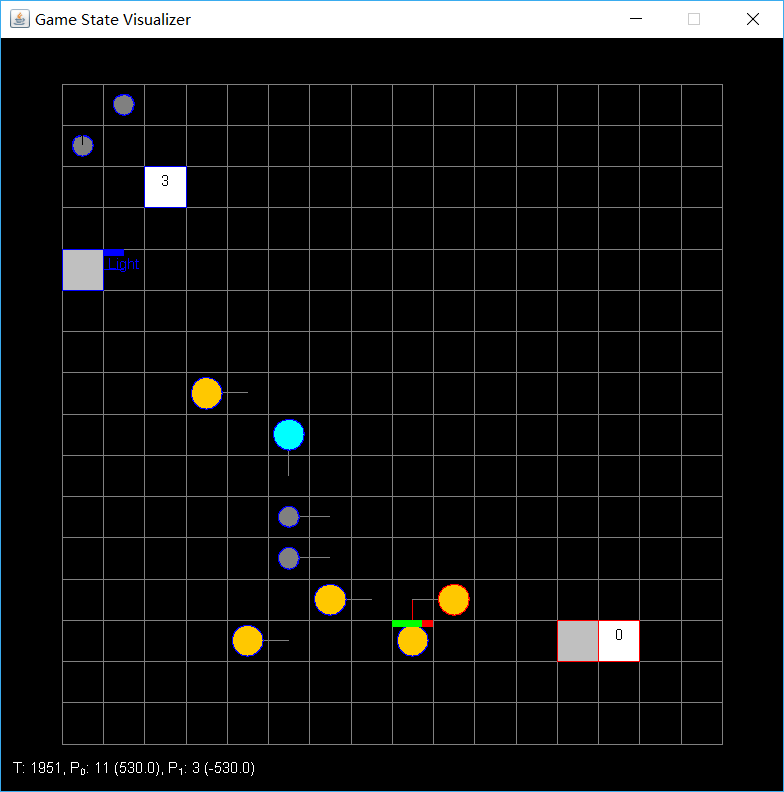


1. 左上 对战HeavyRush

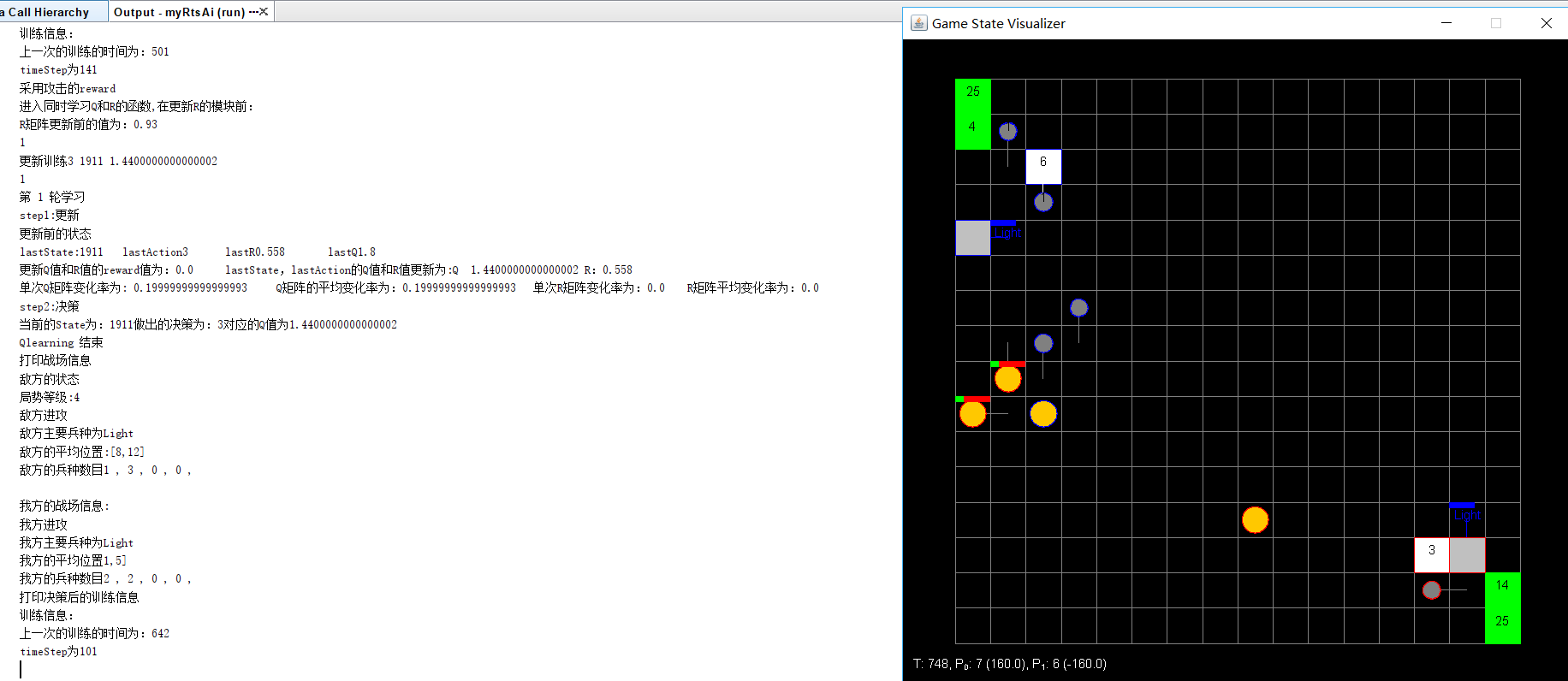


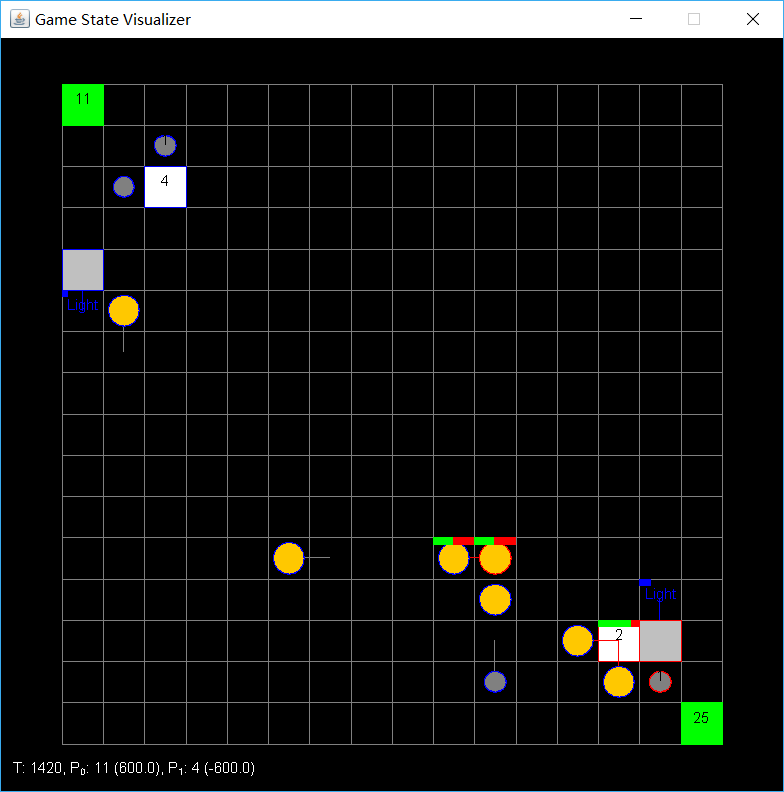
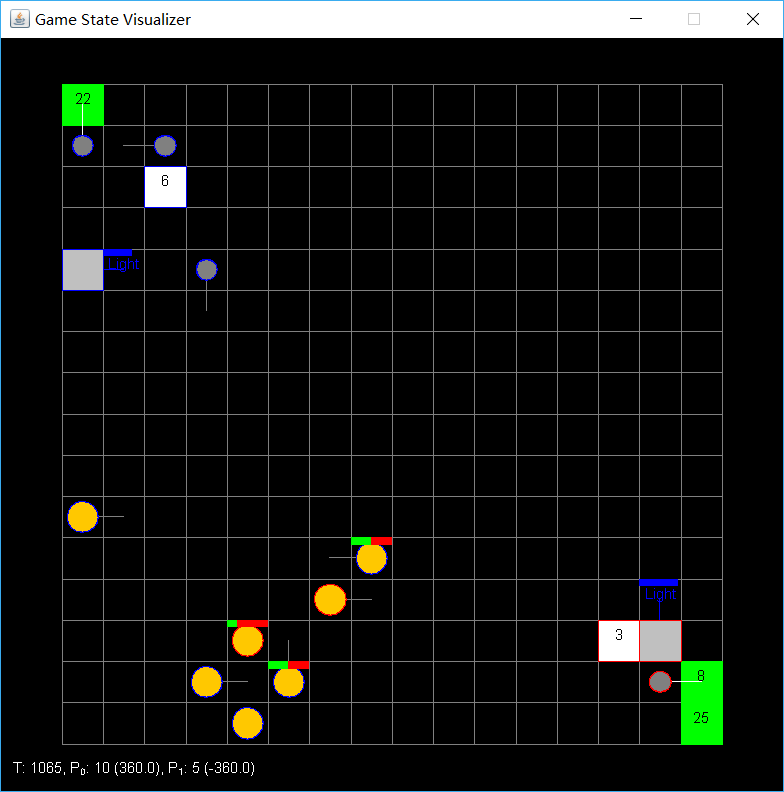
1. 左上 对战同学AI 1号FirstRush



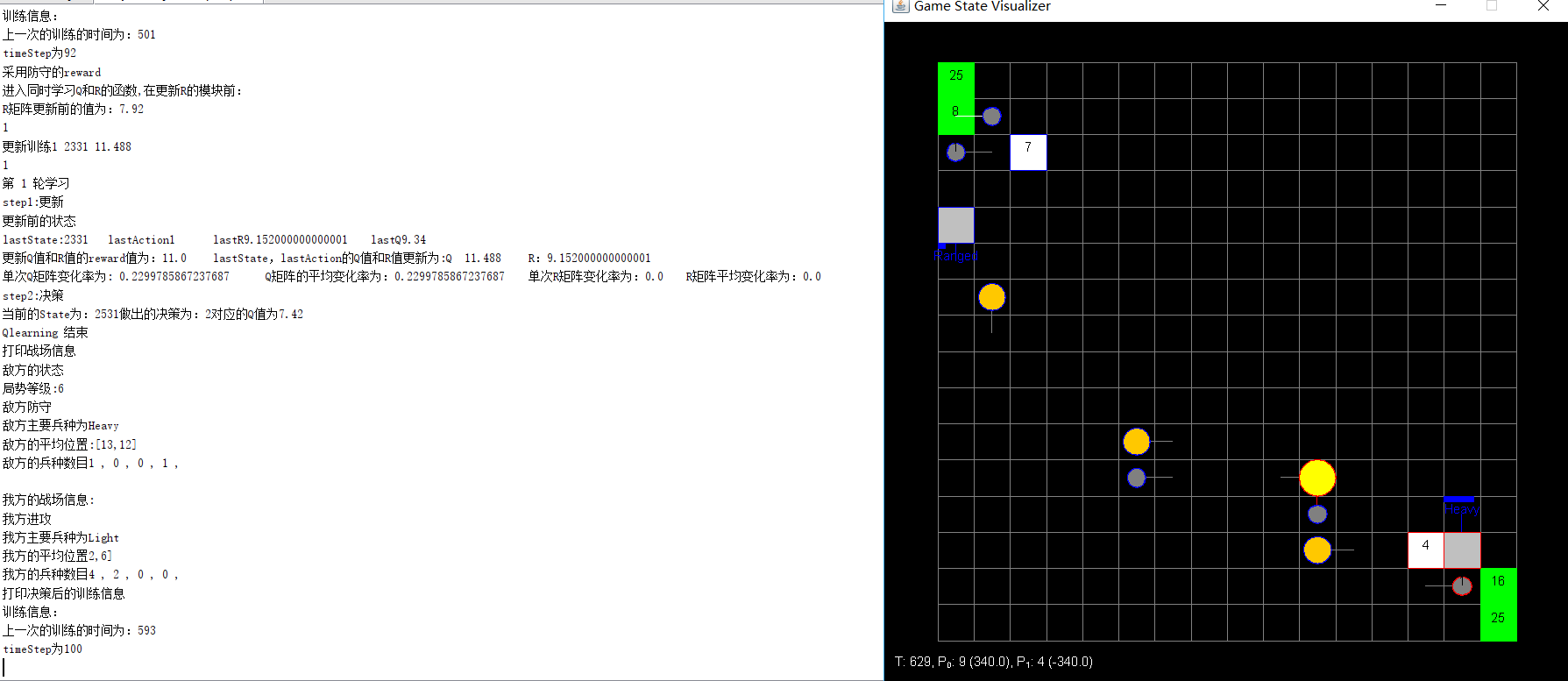


1. 左上 对战LightDefense

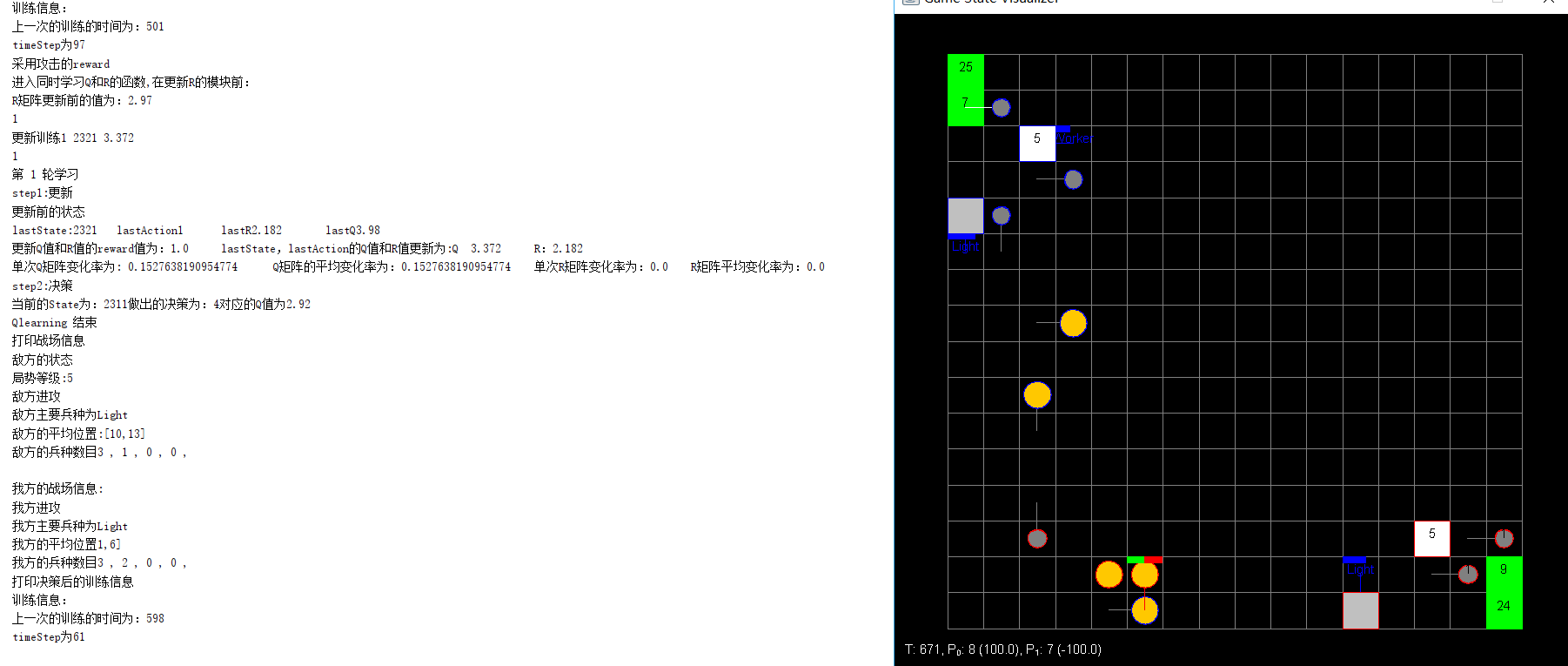


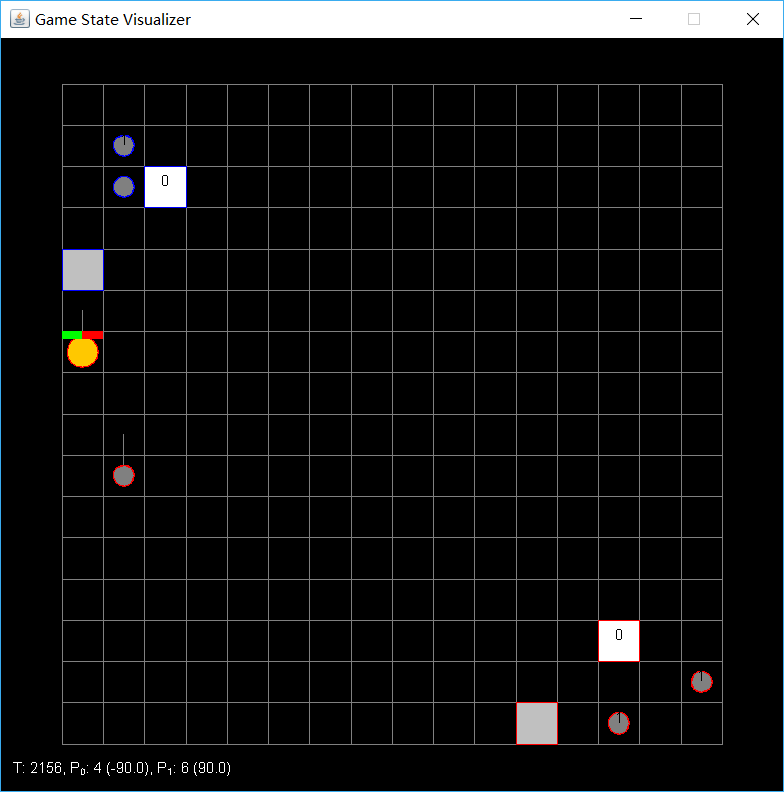
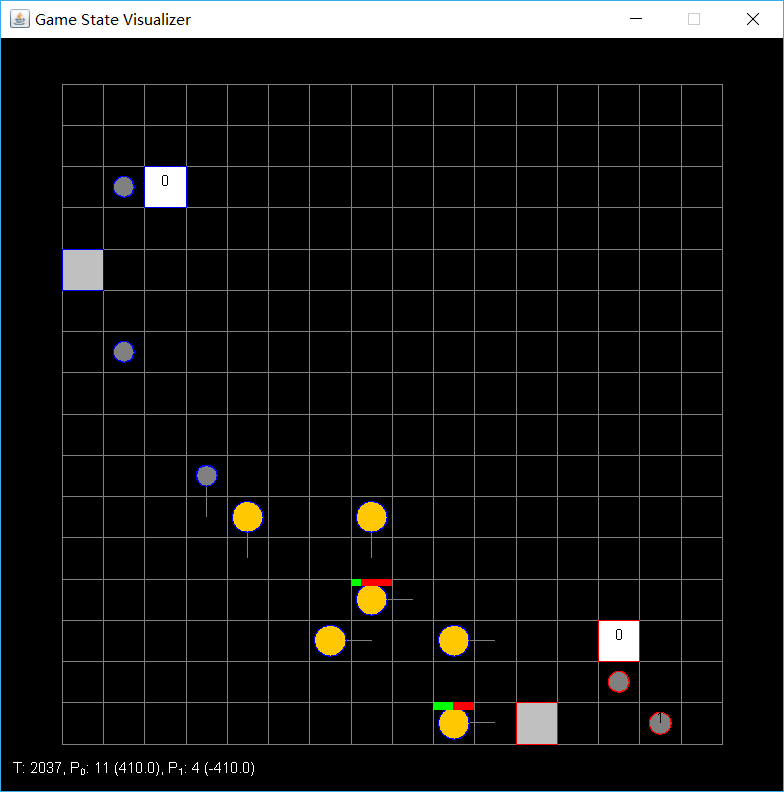


1. 左上 对战HeavyDefense

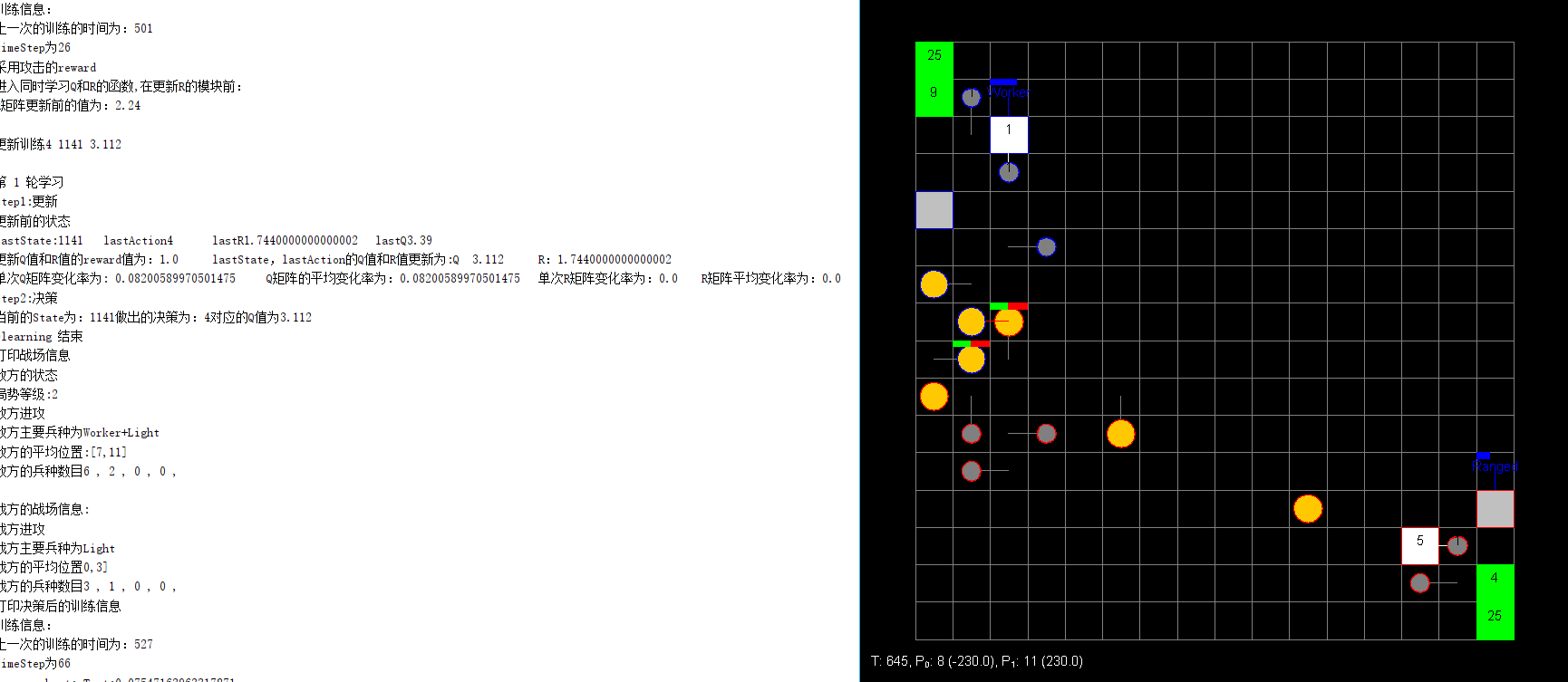


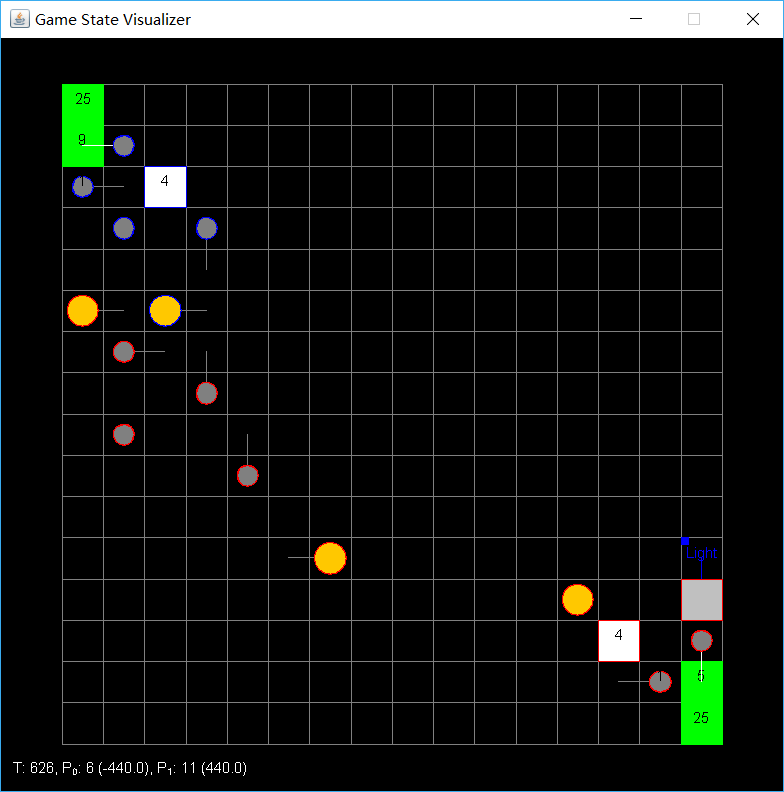
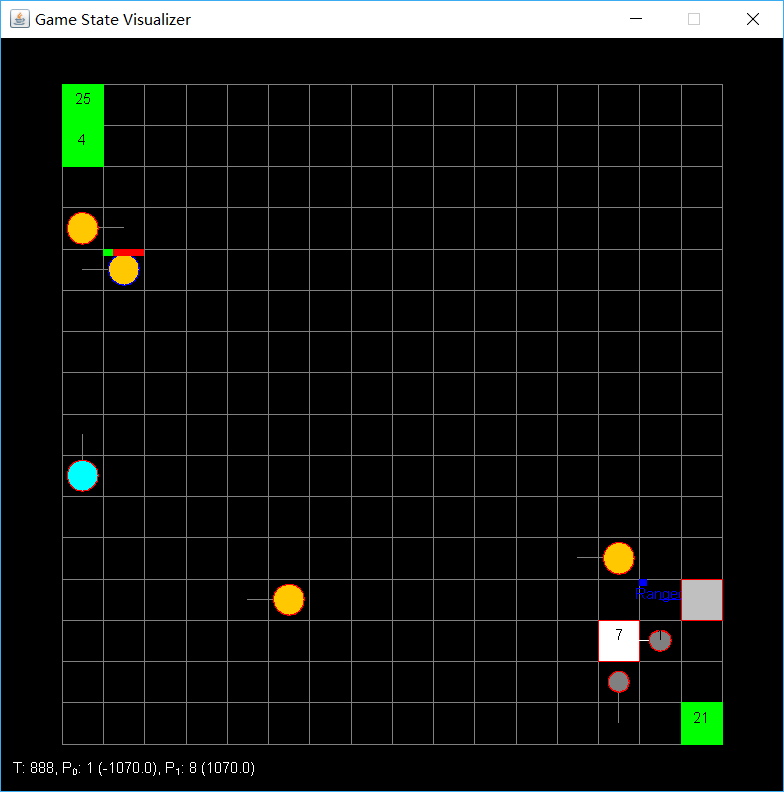
1. 左上 对战同学AI 2号 bottom



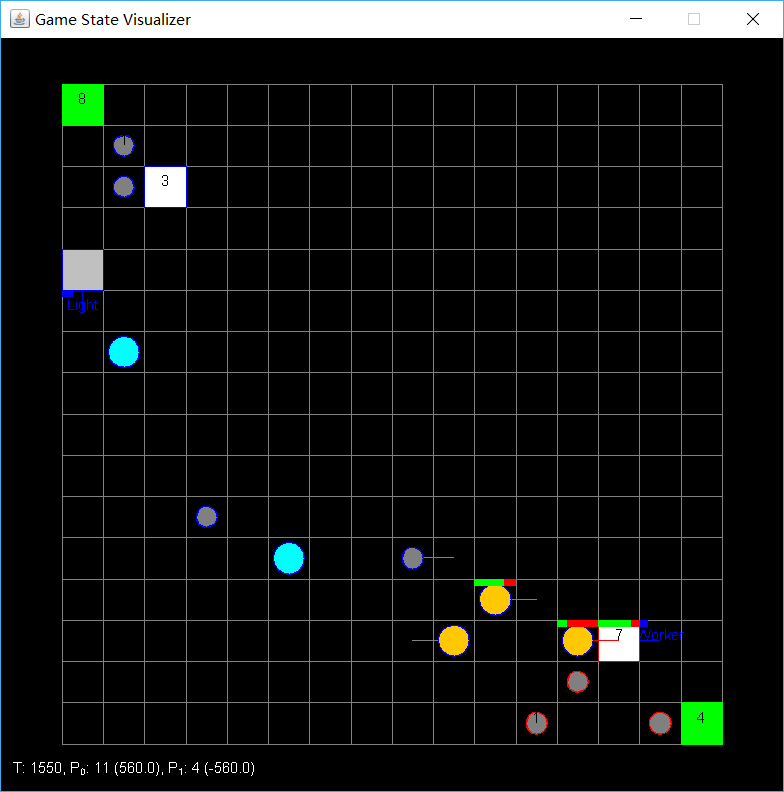
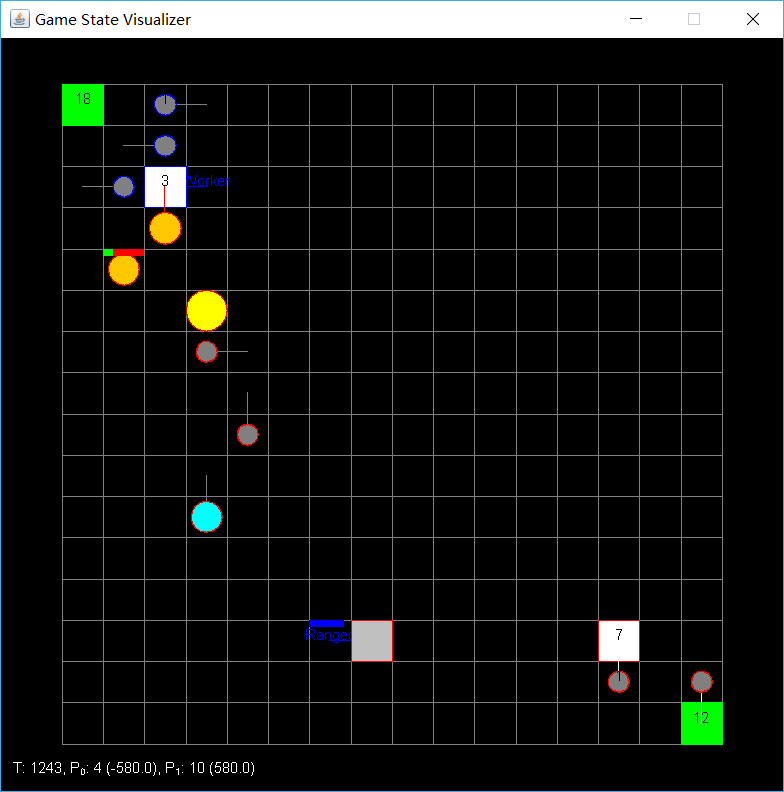
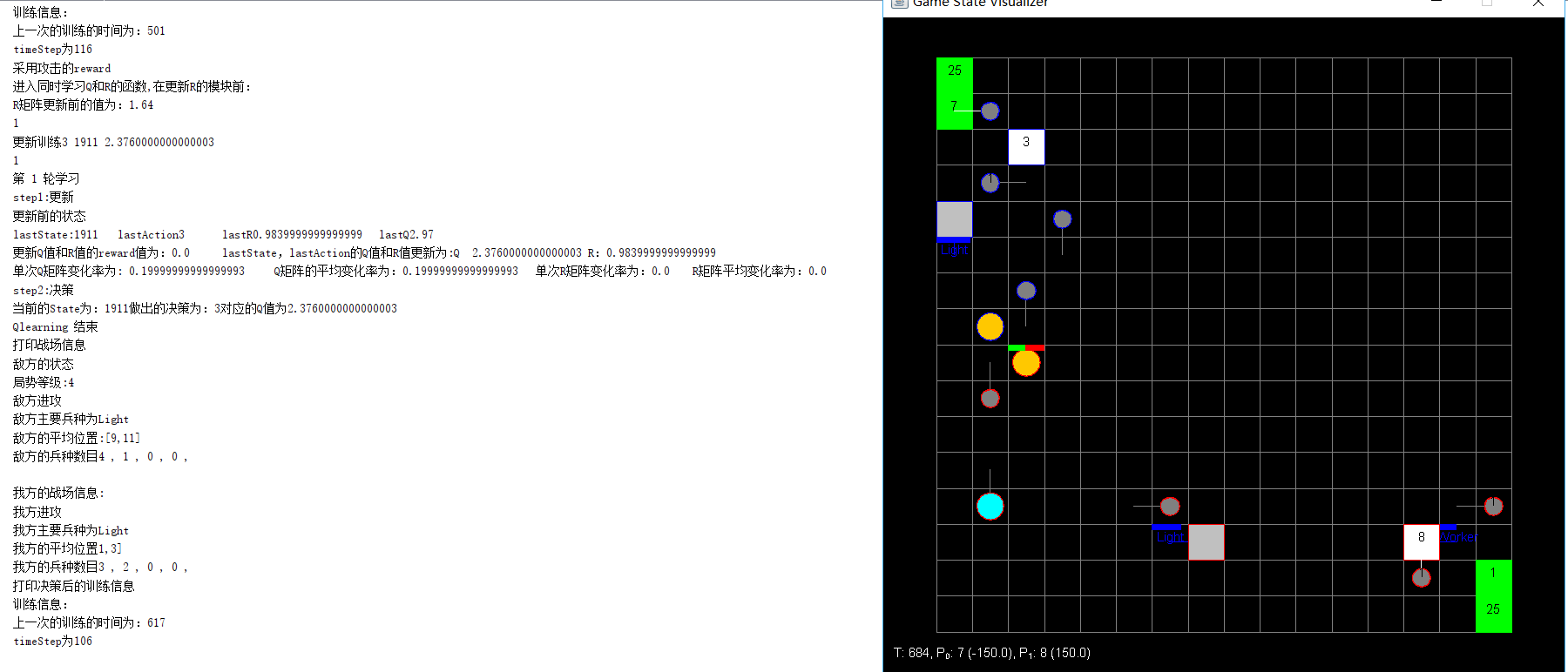


1. 左上 对战同学AI 3号 Simulate



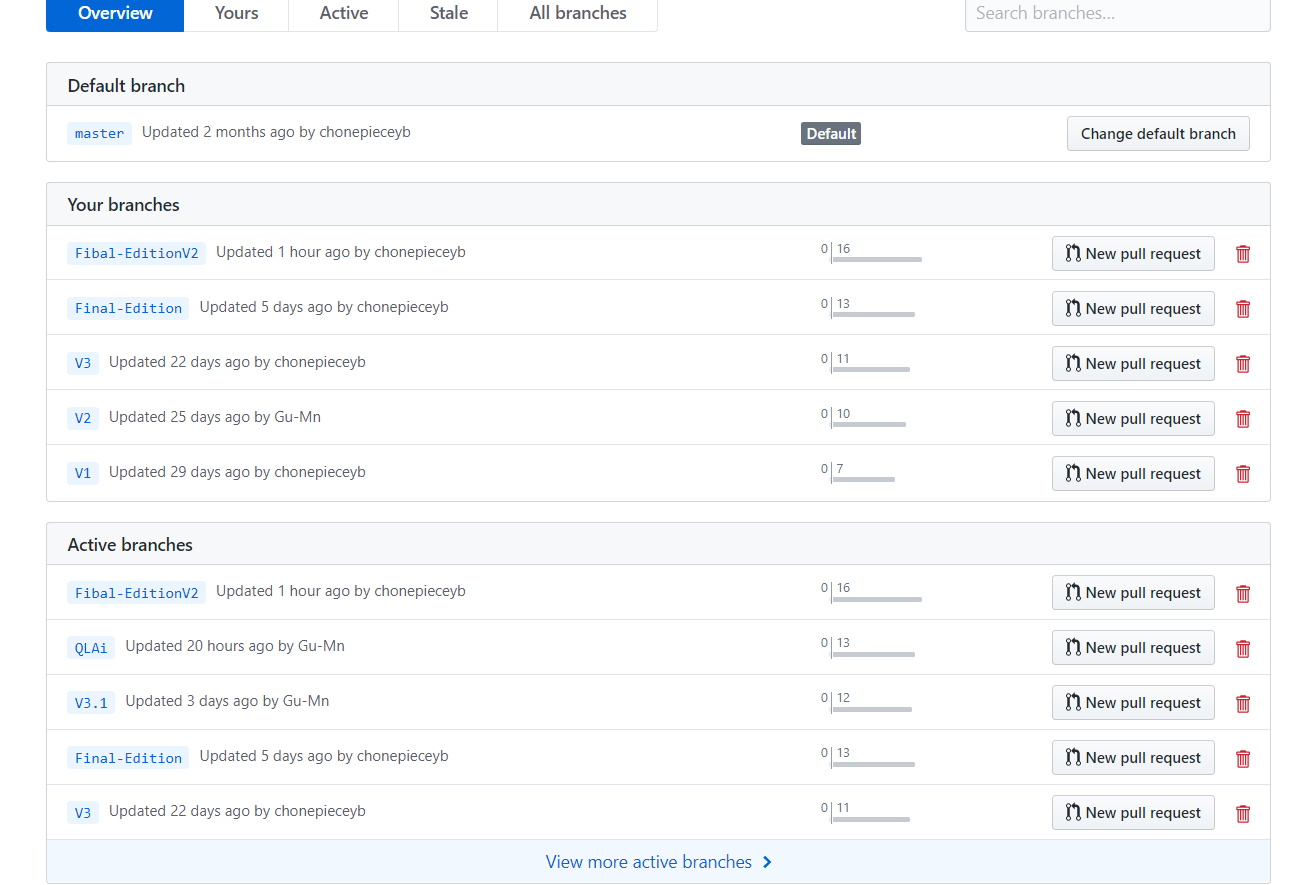


1. 左上 对战同学AI 4号 MyNewAI



**PART III 整个工程的感想和总结**

Rts也写了一学期了吧，从一开始的人工智慧到现在的改进版的Qlearning，我们从这个项目中确实收获良多，初步学习了强化学习的知识，也学会了根据具体的问题来应用算法，或者是通过对问题进行转化（比如我们在AI中将状态离散化）来创造使用算法的条件等等，我们这学期在rts这个平台，针对Qlearning进行了多种优化，提出了多种优化思路，通过具体的实践认识到了Qlearning的局限性我们认为这对之后的进一步的学习能够起到较大的帮助。虽然我们取得了不少的成果，但是还是留下了遗憾，最后的Ai由于时间不足，以及最后几周正好是考试周临近，导致没有调试和训练完成，只能提交改进过的上一个版本的AI进行对战，由于时间原因很多设想还没能来得及实现，比如使用决策树算法（例如C45树，通过采样，对状态进行分类）比如更进一步地采用Deep Qlearning来解决连续的情况，确实是较大的遗憾。但是总体上来说我们仍是收获颇丰整个项目周期将近一学期，在github上建立了6个版本（我们将代码提交到了github托管，等到学期结束将会进行开源）总代码量达到了 ***2000* 行（1500的AI 和 500的QLearning类）,**提升了代码水平，研究能力，学习能力，协作能力。



**(github分支数目)**

**Github源码 ：<https://github.com/chonepieceyb/src.git>**

**[1]** TIM截图20190520204624

（Q-learning 的核心函数，和学习率）