MicroRts第一次实验报告

09017423 杨彬

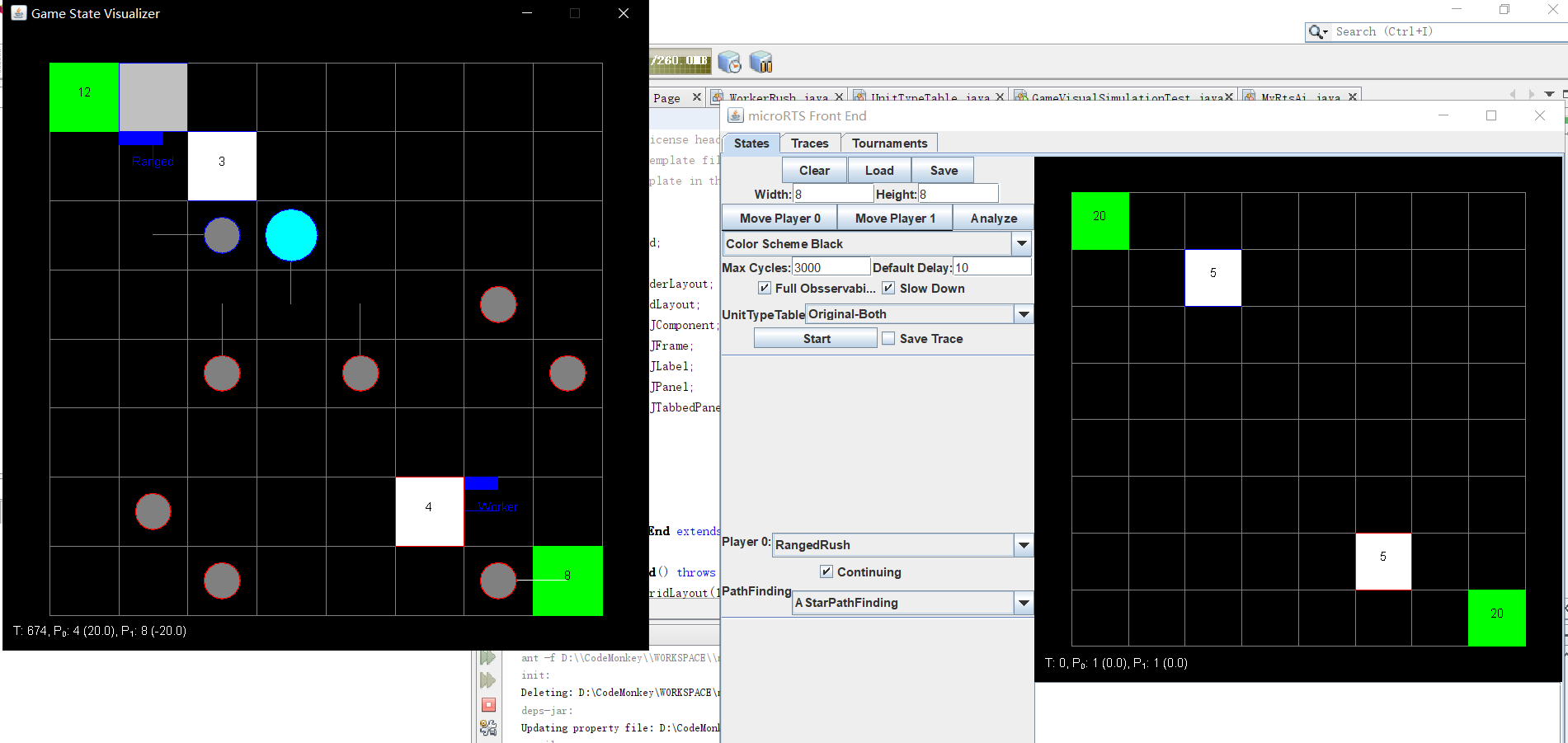
一 实验简介

本学期算法导论课程要求在microrts平台上开发rts游戏的AI，并且对AI进行训练。作为本学习算法实验的第一份报告，这份报告的主要内容包括：1、 microRts平台的部署。2、在microRts平台上部署一个自己的AI 3、对实例AI的代码进行研究，学习microRts的主要API，为以后的开发打下基础. 4根据查阅的资料，对游戏AI的初步理解，以及一些不成熟的畅想。

二 实验内容

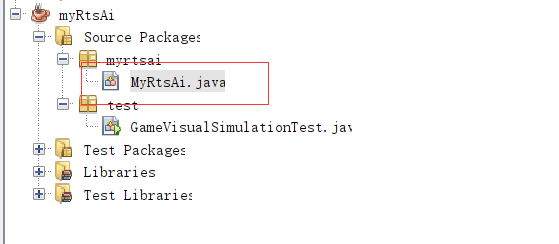
1 microRts平台的部署

从github上下载microRts源码后，在netbeans10 IDE新建JAVA项目，按步骤导入jar包以及microRts的源码，并且成功观看了两个预设AI的对战



2在microRts上新建并且部署了一个自己的AI。

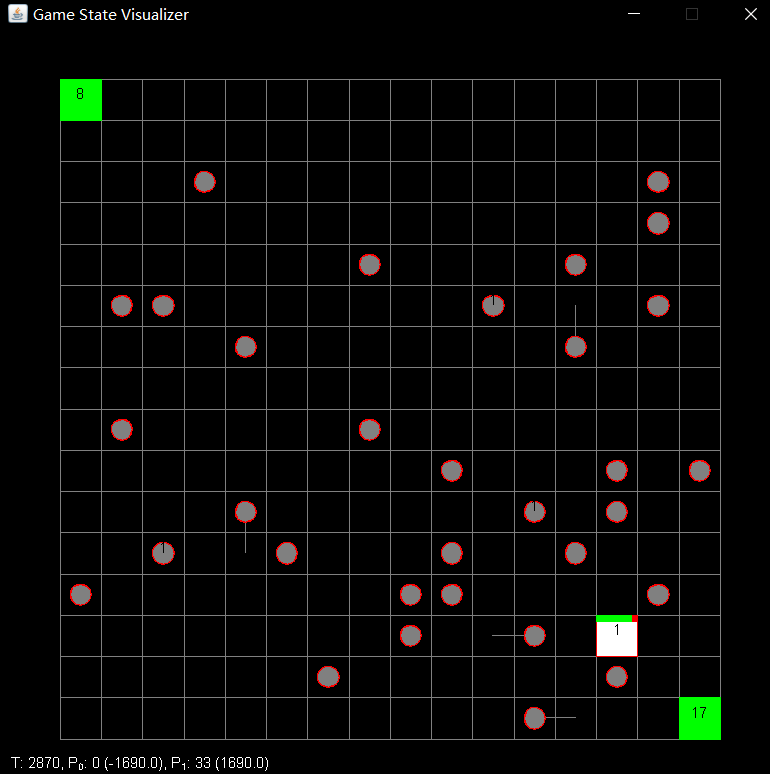
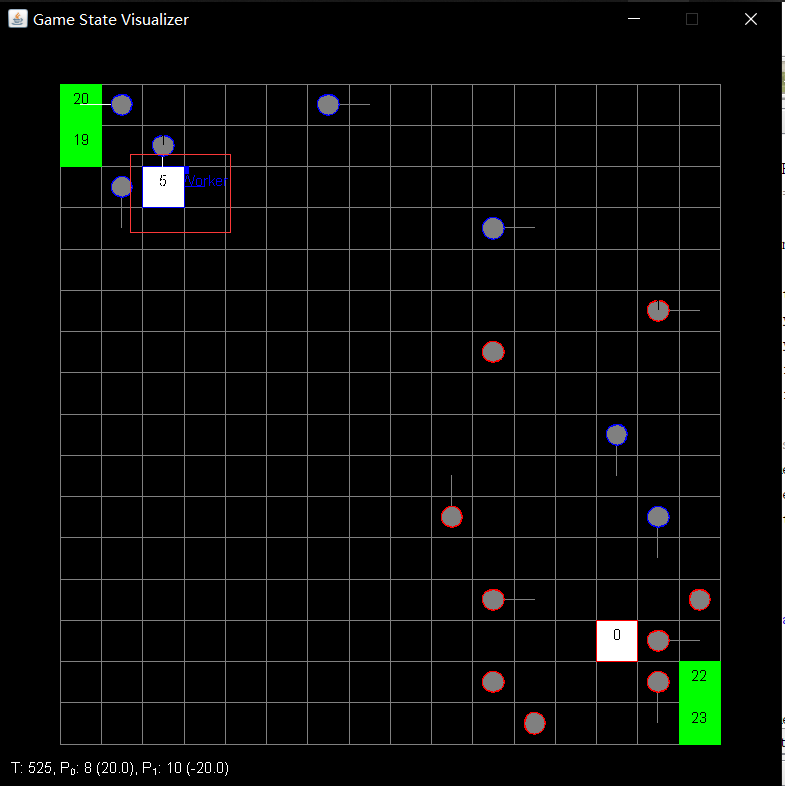
我根据教程，新建了一个Java项目，并且导入了相应的jar，用示例AI(没有灵魂的进攻)，新建了一个自己的AI，并且让自己的AI与预设AI进行了一场对战，结果是显然的，自己的示例AI被电脑AI完虐。





AI项目





AI对决

3、对实例AI的代码进行研究，学习microRts的主要API

1 UnitTypeTable类，这个类里包括了microRts所有的单位兵种的信息，包括每个兵种的ID，建造费用，单位的属性等等。这个类就类似与一个可以随时查阅兵种信息的表，从中可以调用方法，返回兵种的一些属性。

使用范例：

F:\用户\msi-\桌面\新建文件夹\TIM截图20190324211239.png（返回游戏内定义的单位的类型，microRts很多的API需要单位类型作为参数，而单位类型从UnitTypeTable中获得）

 获取训练农民消耗的资源数

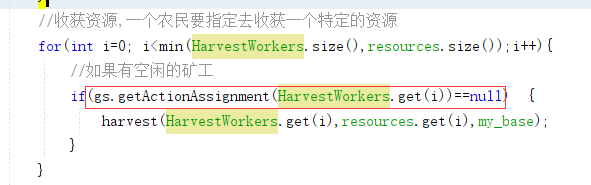
2 PhysicalGameState类，这个类描述了整个游戏的真实的地图信息，例如我们通过这个类来获得战场上存在的单位，并且判断这些单位是否属于玩家，从而，实现了 一种类似于玩家的框选单位的作用

eg:



例如上面的一段代码就是通过 physicalGameState类的实例 pgs的getUnits方法，从战场上获取单位，并通过 unitTypeTable 的getUnitType方法从这些单位中筛选出农民，然后判断农民的ID从而在战场上框选出，所有属于我方的农民。

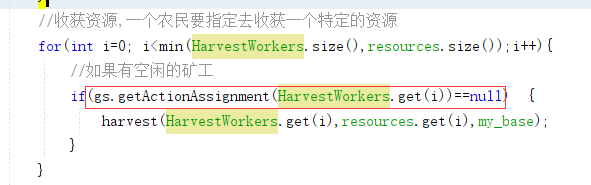
3 GameState类区别于PhysicalGameState类描述的是真实的战场信息，比如说地图的长宽等，这个类描述更加抽象的战场，它接受PhysicalGameState类的实例和unitTypeTable的实例作为参数。通过这个类，我们可以获得许多抽象的战场信息，比如分配给单位的动作，获取单位的动作，以及确定赢家还是输家等等

Eg: 

上面的那段示例代码通过gamestate的getActionAssingnment方法，判断我们所选择单位是否是空闲的，从而从农民的列表中筛选出空闲的矿工，让它们去采矿

4 Unit类，在microRts中，所有的单位都是 Unit类的实例，区别于UnitTypeTable 储存了所有单位的属性，这个类描述了战场上的单位，通过对Unit类的实例的操作，AI指挥单位在战场上作战，而Unit类也是很多方法，比如说基础AI的一些基础函数（训练，进攻）的参数，通过将Unit类的实例作为参数，实现命令的下达。

Eg: 



还是同一段代码，这里的循环变量u 就等同与战场上的一个个单位，而后面的一些函数，eg harvest函数将Unit类的某个实例作为参数，实现让农民去采矿。

5 Player类 这个了类描述了玩家信息

6 abstrationLayerAI类这个类描述了AI的一些基础的抽象动作，比如：训练，攻击，收获资源等等，它作为我们自己的AI的继承的类是十分重要的，我们通过调用它的一些方法，来实现底层的一些基础操作。

Eg: 

这段没有灵魂的进攻的代码，通过attack方法，实现进攻。

三 实验总结与感悟

1 游戏AI开发的初步理解

通过根据给的示例自己创建了一个了一个AI，并且对示例代码的研究，我认为，AI首先是由很多底层的操作构成的，比如说基础的单位的框选，组织农民去采矿等等，这些基本的操作比较固定，因此可以考虑把他们封装成一个个函数，让层次更高的函数去调用。最简单的AI经常是依靠很多条件选择，当满足某些条件的时候，执行某一段代码或者执行某些基础的操作，比如（无灵魂的进攻等）。这样的AI（比如说示例AI）简单但是很僵硬，无法应对复杂的情况。但是AI的开发，其底层应该是有很多，比较固定，比较容易实现比较简单的基础操作组成的。我认为首先应该把这些基础操作封装成一个个函数，写好。而高级的AI则是通过更复杂的条件判断来实现某个基础操作，或者几个甚至很多个基础操作的组合。

通过在网上查阅资料，我目前发现，主流的游戏AI的开发主要有两种模式 1 有限状态机（eg:半条命2） 2行为树。（eg:荒野大镖客救赎）。有限状态机的AI实现起来相对简单，但其本质是通过条件判断，让AI在不同的状态之间转换，不是特别灵活。而行为树则是通过树状结构来实现复杂的条件判断以及决策，虽然行为树是静态的（查阅的资料表明）但是却能实现很多复杂的决策，并且其节点是相对独立的，方便维护，和扩展。

因此，接下来的计划是 1 实现基础的操作（比如 框选特定的单位，组织特定阵型的进攻等等）2 查阅相关的资料和论文，以行为树为基础，实现一个AI demo

2 不成熟的畅想

通过查阅有关行为树的资料时，忽然产生了一个想法（并未证实）。“动态行为树”，行为数的底层是一系列事先定义好的操作，但是其节点，却是一系列动态的节点，在游戏中，我们将一些战场上的信息以一定的权值（比如说，地方单位的数目，地方单位离我方的距离等）量化成为某些变量，而节点有某些参量（eg:阈值）通过变量以及参量来决定节点的返回值，并且在比赛结束后，或者是决策过了一段时间后，根据结果的反馈来修正节点甚至是新增节点，从而实现“行为树的学习”，当局数增多的时候决会变得更加科学，而胜率也会变得更高。