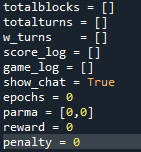
Kevin Charles Hostler

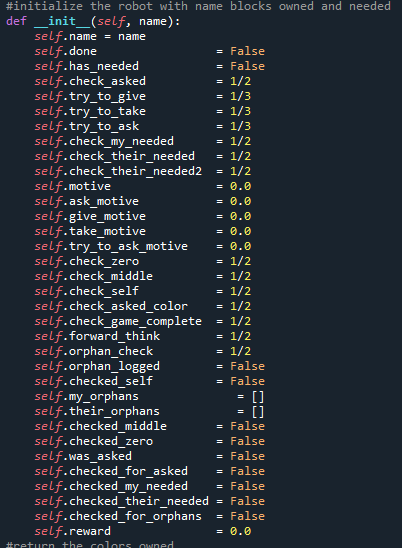
Experimental phase update Version 2:

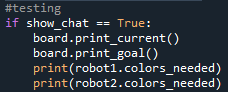
This report is to document and better help explain the plots shown in the PowerPoint slides. To start with we have two different versions of the logic code, though both have a similar premise, but the second version implements fixes to the issues found out in version one.

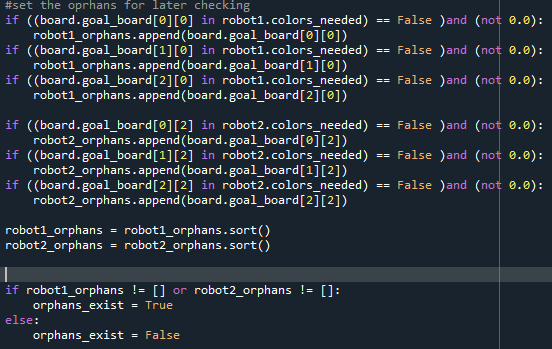
**Version 2**

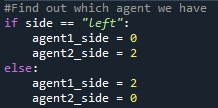
This logic version was created and edited with several key ideas in mind. To start with version 1 would end up with the agents getting stuck for several turns when blocks would fill up the middle or they would get the blocks they need but leave blocks neither need but one needs to take in the center. So this boils down to the version 1 agents not being able to take actions that have future rewards as in completing the game before turn time-out.

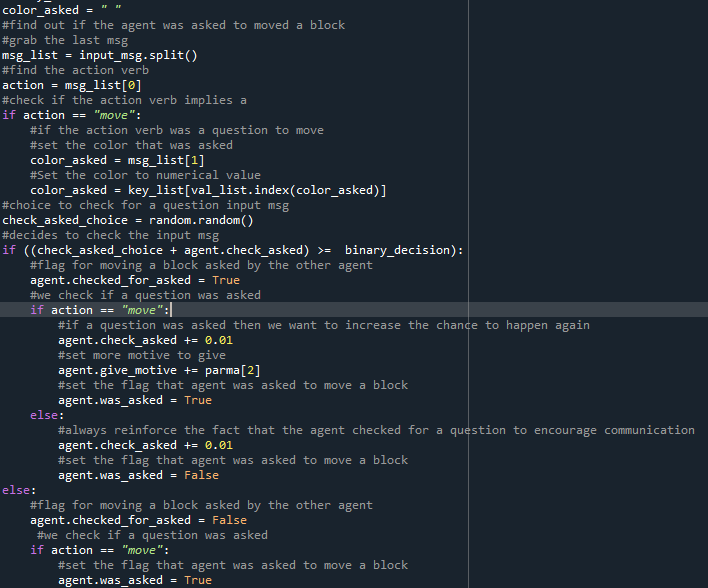
The first thing we do is set all of the variables that are tracked each game to global to be used for the graphs at the end.

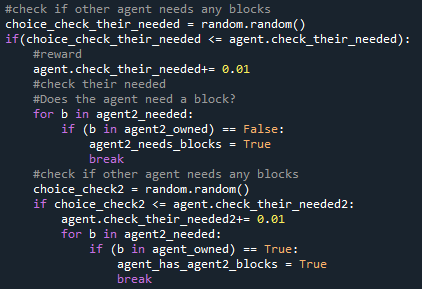
 The new additions to each agent are the decision chances for forward thinking checking, orphan block checking, game completion checking and orphan logging.

 The second major change to help the optimization time of training was limiting the i/o of the print statements and have moved all communicating to internal lists, though for testing the messages can be printed again by a simple Boolean switch.

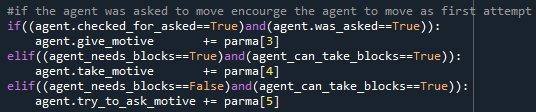
 The next major change is the addition of an orphan verification system where before the agent and game loops begin we verify which blocks are never needed and if they need to be moved because of it. This will be used by agents during decision making.

Set the agent coordinates so that we don’t do this in the decision loop.

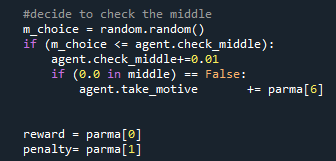
 The check for a question logic is mostly the same, though this time the rewards have change and can be altered in loops for testing. The agent also keeps better track of what occurred for further decision help.

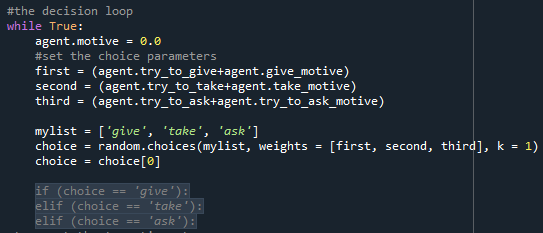
The next major change was to have the agent decide to check if the other agent needed any blocks outside of the decision loop. This greatly helped to reduce the training time.

 This new logic does a great deal of heavy lifting as it is the forward thinking decision. This has the agent decide if it wants to check for possible ways to end the game. It finds out if it doesn’t need to take blocks but should, then what those blocks that aren’t needed but should be moved, and where who they should belong to.

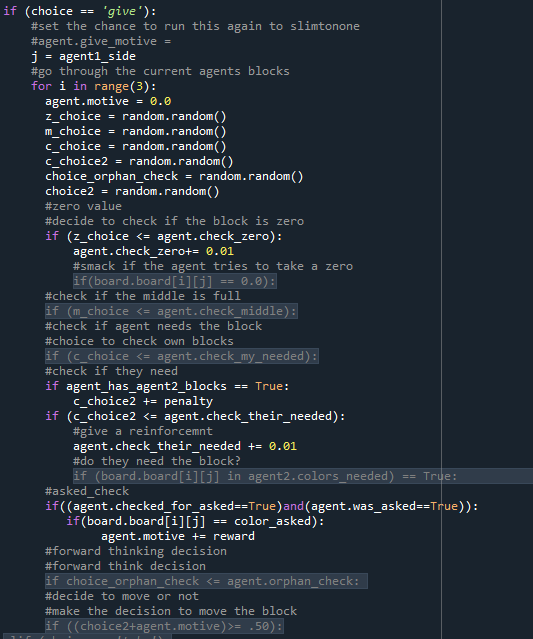


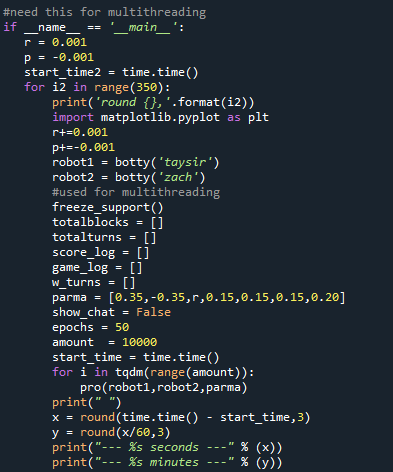
After the game completion check, the agent gets motive increase for it’s first action based on what was found out.

The last two things we do before the decision loop is set the current reward and penalty amounts and have the agent decide if it wants to preemptively check the middle.

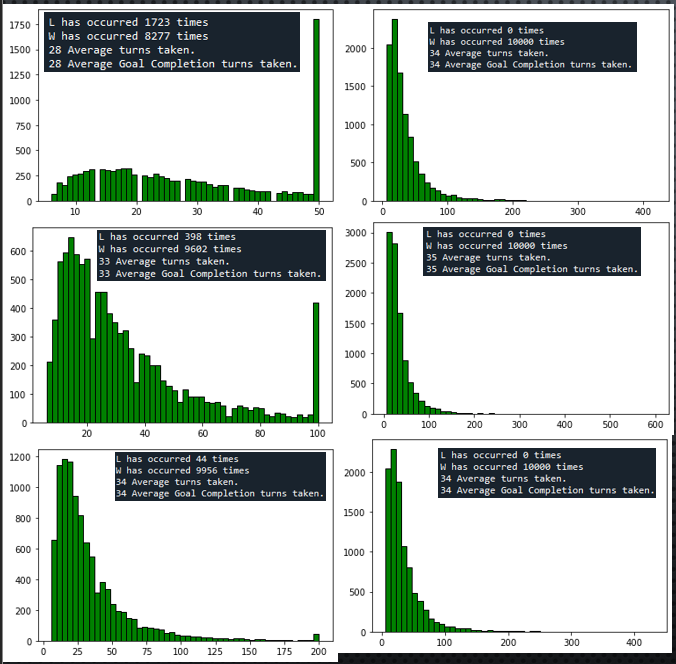


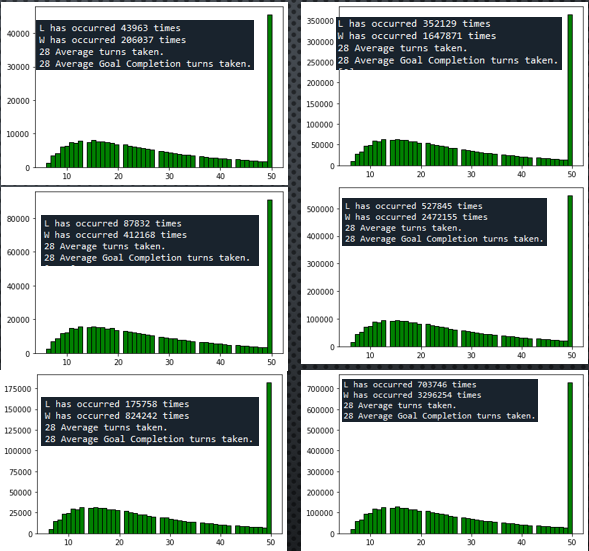
Another major change for the better was to use pythons random sub list function in place of the previous method for chance decisions, this enabled be to set how likely one option was over another and made for better results with increased/decreased motives.

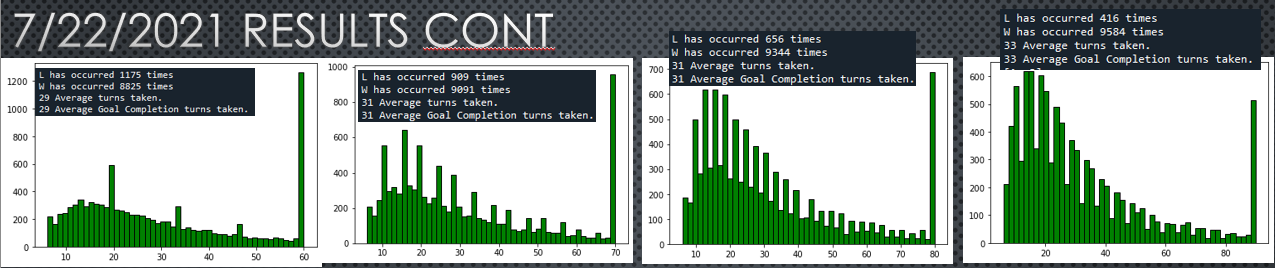
The actions again all follow a similar state, all random numbers are obtained at the beginning to ease the time of iteration, the agent then decides to check if the current block is empty, in it’s own needed, the other agents needed, an orphan or if the middle is full.

 This is were the new code differs, since we streamlined the parameter set up to be inside of the list parma, we can now change them in loops to see effects the model. Several things of note. All of the motives for the first action to be picked are higher, the rewards and penalties for actions committed to are much higher than version1 as well. As we will see in the graphs this is a very important change.

Results



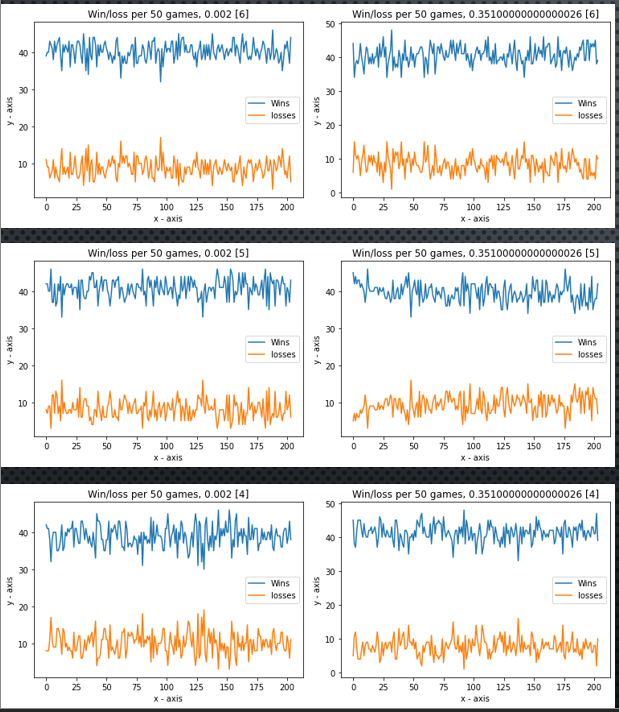


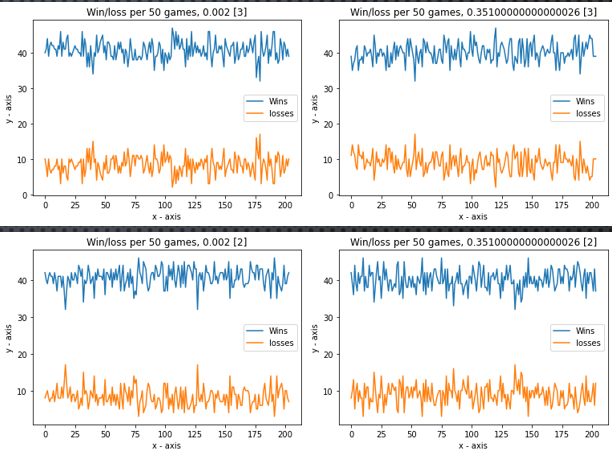


Summary:

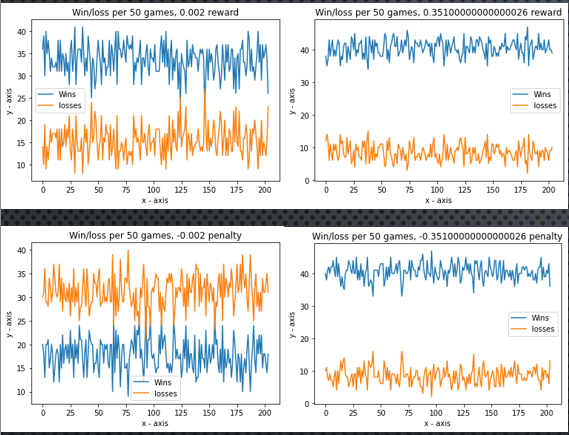
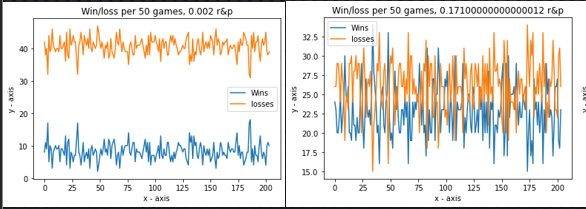
As shown by the graphs the new logic has fixed several issues that the old logic struggled massively with. To start, the implementation of higher rewards and penalties greatly assisted the agents abilities to select proper actions per states but the most definite change would be the forward thinking check. At first the agents don’t think about orphans but as the games progress and they accumulate more rewards, the begin to “learn” that they need to check for which blocks should be moved even if it has no immediate value to them, as completing the game is the goal.

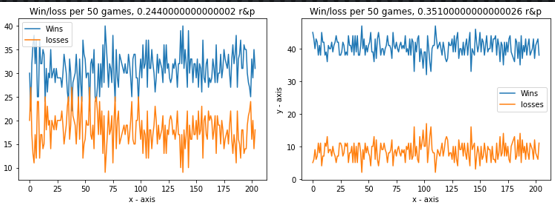
Even the 50-game time out yields massive improvements, in version 1 it was around 50% win/loss at any game amount. But now even at 10k games the agents manage about %80 win ratio.





[2] to [6] are variables that dictate the first action ie, chance of move takes or ask attempt as a first try. I ran tests on each to range each variable chance from 0.002 to 0.35 in 0.001 intervals. From my testing it seems that while higher intervals help the win amounts it doesn’t make or break the overall averages.  
The graphs show number of wins and losses per 50 games for 10k games





The above graphs show altering the values of reward only, penalty only and both at the same time. While we can observe that the reward being much lower does affect the win/loss ratio, it doesn’t compart to the flipped starting games when rewards is high and penalties is low.

The extensive testing on 10k games has shown that the RL model seems to listen better when it **knows what not to do** rather than **what to do.**