Student Name  
Nathan Korzekwa

Class   
CS3630

Professor  
Frank the Tank

Homework 2: Reinforcement Learning

**Section 2\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*Article 1: Implementation Choices*

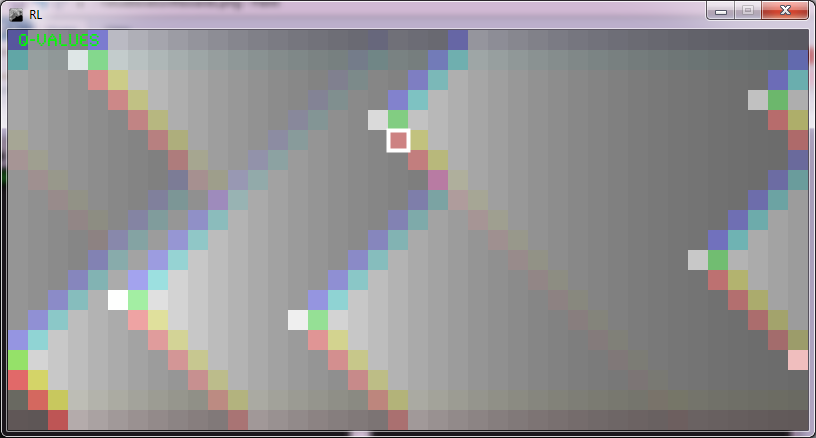
Value iteration is a simple algorithm for finding the true value of each state (although it can take a while to converge), and as such, my implementation was simple. At each step in the learning process, one need only loop through all the states and possible actions to determine the Q-values and the true values at each state. Since the implementation in the PDF called for only one step in the discount process, all the values of nearby states have only γ as a discount factor. I left gamma at the default 0.9 for this assignment.

*Article 2: Screenshots*

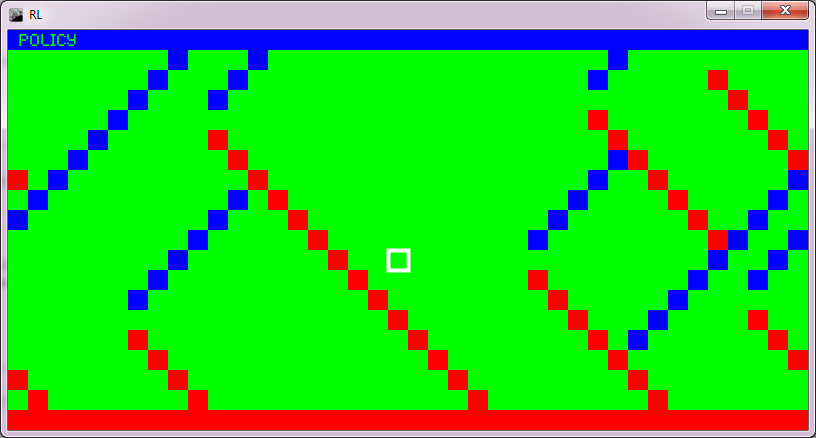
Rewards:



Values near Convergence



Generated Policy



*Article 3: Evolution of the Value Function*

The values of the states at the first iteration are simply the immediate rewards at each state. After time, the values propagate out in different directions, away from the point of immediate reward, decaying at a rate specified by where *i* is the iteration. Intuitively, the values are higher near a point of immediate reward, and lower further away.

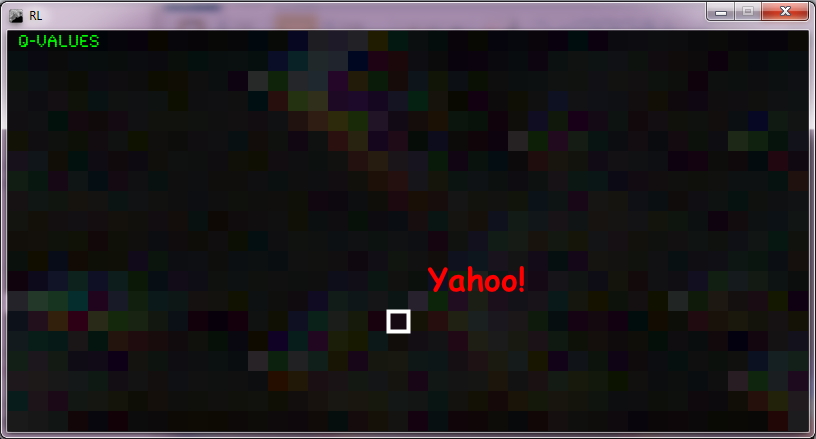
**Section 3\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*Article 1: Implementation Choices*

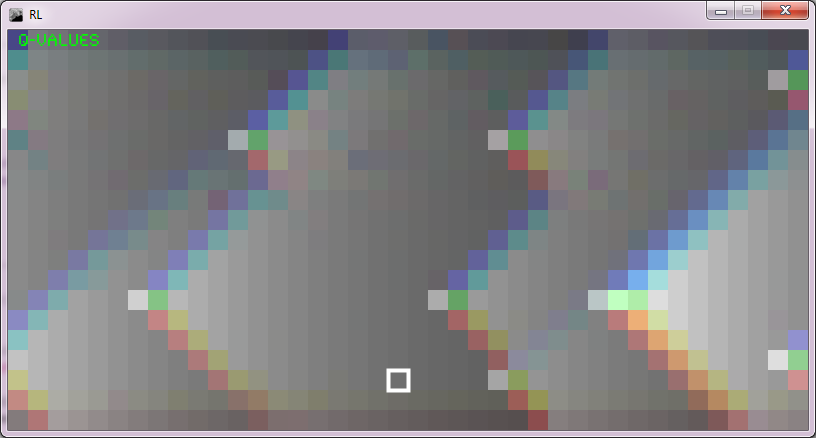
Implementation for Q-Learning is even simpler than the implementation for Value Iteration since each iteration is driven by an action taken in the world, so only one update step need to be taken at a time. I found that a learning rate α of 0.8 was a good choice for a balanced learning process. I also had to keep the “maxV” variable up to date with the process of Q-Learning, so that values on the screen were scaled properly to the 255 range.

*Article 2: Screenshots*

Far From Convergence



Near Convergence



*Article 3: Q-Learning vs. Value Iteration*

Q-Learning is, at heart, very similar to Value-Iteration, except with the added restriction that we do not know the transition function, and that we do not immediately know the initial rewards of each state. As a result, the world state-space must be explored to get the initial rewards associated with each state, and to get the values of the states nearby the rewards.

Since, for this assignment, we employed a random exploration strategy, the convergence of Q-Learning was especially slow, especially in comparison to the more informed and orderly Value Iteration.

*Article 4: The Utility of Exploration*

Without any exploration at all, Q-Learning would be rendered completely useless, as the agent or robot initially knows nothing of the world. With exploration restricted to the current policy generated by Q-Learning, the agent would get stuck in a localized area, and it would be difficult, or in some cases impossible, to break out of. To ensure knowledge of all the rewards and the best paths to them, a full exploration of the world must be possible. For this assignment, we used random searching, which takes a while and is not guaranteed to provide convergence in a finite amount of time, but, pragmatically, it gets the job done.