# Reinforcement Learning

[Overview](http://docs.google.com/index.html)

[create your own maze](http://docs.google.com/maze_editor.html)

[Value Iteration](http://docs.google.com/ValueIter.html)

[Policy Iteration](http://docs.google.com/PolicyIter.html)

[Q-Learning](http://docs.google.com/QLearning.html)

[Prioritized Sweeping](http://docs.google.com/PSweeping.html)

Installation and Execution:

This tool is know to work with Java Runtime Environment(JRE)1.4.2 and above. To install JRE1.4.2 and above visit <http://java.sun.com>.

Once java is installed and it is in path:

1. Extract RL\_sim.zip to appropriate directory.
2. On windows: Start command prompt,
3. On Linux: Start shell.
4. Change directory to directory where files are extracted. Then Change directory to go in RL\_sim directory.
5. Execute the command 'java -jar rl\_sim.jar'

To create a shortcut on windows:

1. Right click on desktop, select new, select Shortcut.
2. Copy command 'java -jar rl\_sim.jar' as location of item and click next.
3. Specify RLSim as name of shortcut and click finish.
4. Right click on the shortcut and select properties.
5. In Start in box, specify the absolute path of directory in which RL\_sim.jar exists.
6. Press Apply and Ok. The shortcut is ready for use :)

If you want to recompile and execute from the source code the main class is called MainUI.java

### Rules of the Game

The experimental setup consists of an agent moving in a discrete state space represented by a maze where each state is represented by a cell in the maze. The maze contains terminal states represented by goal states and obstacles represented by walls. The maze is bounded on all sides by walls. If the agent tries to transition from one state to another and hits a wall instead then the agent receives a positive penalty and stays in the same state. There is a path cost of 1 unit associated with every transition that the agent makes from one state to another. The aim of the agent is to find that path to the goal state which has least cost associated with it.

To model the noise in the environment a parameter named ‘pjog’ is used. Each state has a finite number of successors, N. If in a particular state s the agent decides to perform action a then the agent will end up in the valid successor of s with a probability equal to (1-pjog) and end up in any one of the N-1 successors of that state with a probability equal to pjog/ (N-1).

For Q learning and prioritized sweeping another parameter, called ε, is used. This is specifically to implement the ε-greedy policy. Under this policy the agent decides to perform the best action with a probability of (1- ε) and performs any random action with a probability equal to ε/(N-1).

### Credits

This tool has been developed by [Rohit Kelkar](http://rohit.freeshell.org/) and [Vivek Mehta](http://vivekm.freeshell.org/), as part of the Extended Course Project for MS in Information Technology with specialization in Robotics Technology, at [Robotics Institute](http://www.ri.cmu.edu), [Carnegie Mellon University](http://www.cmu.edu).

**Advisor:** [**Prof. Andrew Moore**](http://www-2.cs.cmu.edu/~awm/)

### Contact

For any query regarding this tool, send us an email.

[Rohit Kelkar](http://rohit.freeshell.org/): rohitkelkar28 [AT] yahoo [DOT] com

[Vivek Mehta](http://vivekm.freeshell.org/): vivekm [AT] gmail [DOT] com