1. The MDP we have created is a basic version of Mario in which each state has different reward points. We have created 10 states in this case but it can be easily extended to any number of states. At each state the following actions are valid T(top), B(bottom), J(jump), R(Reverse Jump). Top goes to the next state with 0.9 probability and stays at the same state with 0.1 probability. Bottom goes to the previous state with 0.9 probability and the same state with 0.1 probability. Jump goes to the next to next state with 0.9 probability and the next state with 0.1 probability and Reverse Jump does the same in opposite direction. The rewards are generated randomly for each state between -2 to 8. The last state has a reward of 20 points. If at the bordering state the action results in remaining in the same state.
2. The Value iteration and policy iteration gives the policy at each state and also the utility of each policy. We can clearly see the correct policy corresponding to the rewards at each stage. We can also compare the policies using the policy iteration and value iteration and see the differences in the utilities in case they are different. We can improve this implementation to include other type of actions. We can also modify reward points dynamically for a state. We can also have different opponents like in the Cat-Mouse example. We would also like to implement a graphical user interface similar to cat-mouse example.
3. The MDP method is suitable here because it brings uncertainty to the world, which makes it more game like. Each action has a certain probability to go wrong and the optimal policy has to be decided keeping that in consideration.