Intro:

The reinforcement learning is one of the fundamental aspects of the artificial Intelligent in games. This project is to implementing reinforcement Learning algorithms on Minecraft and analysis the training result of each bot. The implemented control algorithms used MARLO as a framework. MARLO is a library built on top of Microsoft’s Malmo project which based on Minecraft(Microsoft, 几几年). The three control algorithms are based on Monte Carlo Tree Search, Q-learning and Deep Q-learning.

The aim of this report is to explain each of algorithms separately and analysis the summaries.

Algorithms:

1. Monte Carlo Tree Search Bot:

Monte Carlo Tree Search is a heuristic search algorithm. The algorithms of this bot is based on the Monte Carlo Tree Search.

In Monte Carlo Tree Search, there are 4 basic steps: Selection, Expansion, Simulation and Backpropagation.

In this bot, the selection steps were implemented as a calculate action (calAction) function. The required information to calculate action is the location of the character (X, Y and Z value), reward that can directly obtained from the MARLO framework, and previous action that stored in the global variable. The function Calculate action taking actions from the action pool.

The Expansion steps were implemented as that the parent action perform a child action that calculate from the calculate action (calAction) function.

1. The required information is location of the character, action and reward, which is the value and reward. The reward can get directly from the MARLO framework
2. Each step
3. The function Calculate action taking actions from the action pool(3, 7, 8) and add the taken action to the child action. child action is the next step The using [UCB1] (select action Algorithms)
4. and using update value function(UVF) to calculate a value and update the value
5. The action can only go forward left and right, so the length need less than 30
6. Q-learning bot

Q-learning is a reinforcement learning algorithm. In Q-learning Q is the action-utility function which used to evaluate the pros and cons of taking an action in a particular status. The combination of status and action is limited. The Q(action-utility function) can be taken as a table that each row in the table records the status (Delta x, Delta z, Yaw), and rewards when selecting different actions (go forward, left, right).The table is initialized to zero, then each row is updated by rewards through training.

1. Deep Q-learning bot

It is obviously unrealistic to maintain a large Q table by the traditional Q-learning method due to the excessive state. However, deep Q-learning is a model-free and off-policy reinforcement learning algorithm which solves the reinforcement learning task by playing games in the emulator following an episodes-greedy policy forexploration of the search space. To approximate the value of the Q-table, neural networks train the actual training sample data.