# TRAIN A REINFORCEMENT LEARNING AGENT TO PLAY MARIO

**Group#6, Jiajun Ren, Zhouyang Wang, Yuen Zhou**

## Our goal:

As a classic game on NES, Super Mario is one of the most famous game series in the world. In this project, we tried to use Double-Q learning to train a Mario agent to play the game by itself and try to pass the game level in its shortest time.

## State space:

1. Mario
    a. standing
    b. running
    c. jumping
    d. dying
    e. resurrecting
    f. eating props
    g. using props
2. Monsters
    a. standing
    b. moving
    c. dying
3. Props
    a. hiding
    b. appearing
    c. being eaten

## Specification of the actions:

Mario has several different movement styles, including staying still(refers to standing), walking left and right(running), jumping in place, and jumping right, big, or combo jumps(jumping). If he encounters a monster or falls off the map or the time runs out, Mario will die and resurrect at the beginning. when he touches the props, if it’s a coin, he could use the prop, and other items would be stored and could be used in the future.

## Reward:

The rewards after every action are composed and decided by three variables: death, time, and the agent's position. Reward = agent's position value + time value + death, and it is in the range (-15,15). For the agent's position value, the agent's position value = x1(agent’s x-axis position after the action) - x0(agent’s x-axis position before the action). For the time value, time value(c) = c0- c1, while c0 is the clock reading before the action and c1 is the clock reading after the action. In that case, the penalty will be implemented when the agent is standing still. For death, if the agent(Mario) is alive, the death = 0, and if the agent is dead, the penalty will be implemented, the death = -15.

**Techniques description: Double Q-learning**

Consider the target Q-value for Q-Learning in general:

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}}\left[r + \gamma \ \max_{a'} Q^*(s',a') \middle|\ s,a\right]$$

$$max_{a'} Q^*(s',a')$$

The use of maximum overestimation is an implicit adoption of the estimate of the maximum value itself. This systematic overestimation introduces maximization bias in learning. Since Q-learning involves bootstrapping - learning estimates from estimates - such overestimation can be problematic. Instead, the double q-learning solution involves the use of two separate Q-value estimators, each used to update the other. Using these independent estimators, we can perform unbiased Q-value estimation for actions selected using opposite estimators. Thus, we can avoid maximizing bias by separating the updates from the biased estimates.

**Expected results**:

Pass the game within the limited time and try with the shortest time.

**Number of Hours Necessary:**

We expected the project to take around 4-5 weeks and 120 hours(40 hours/person) to complete.