



# 강화학습으로 게임하기

김승우 김정학 송서영 정하영 조준익 임규철 정영만



미쳤습니까 휴먼?

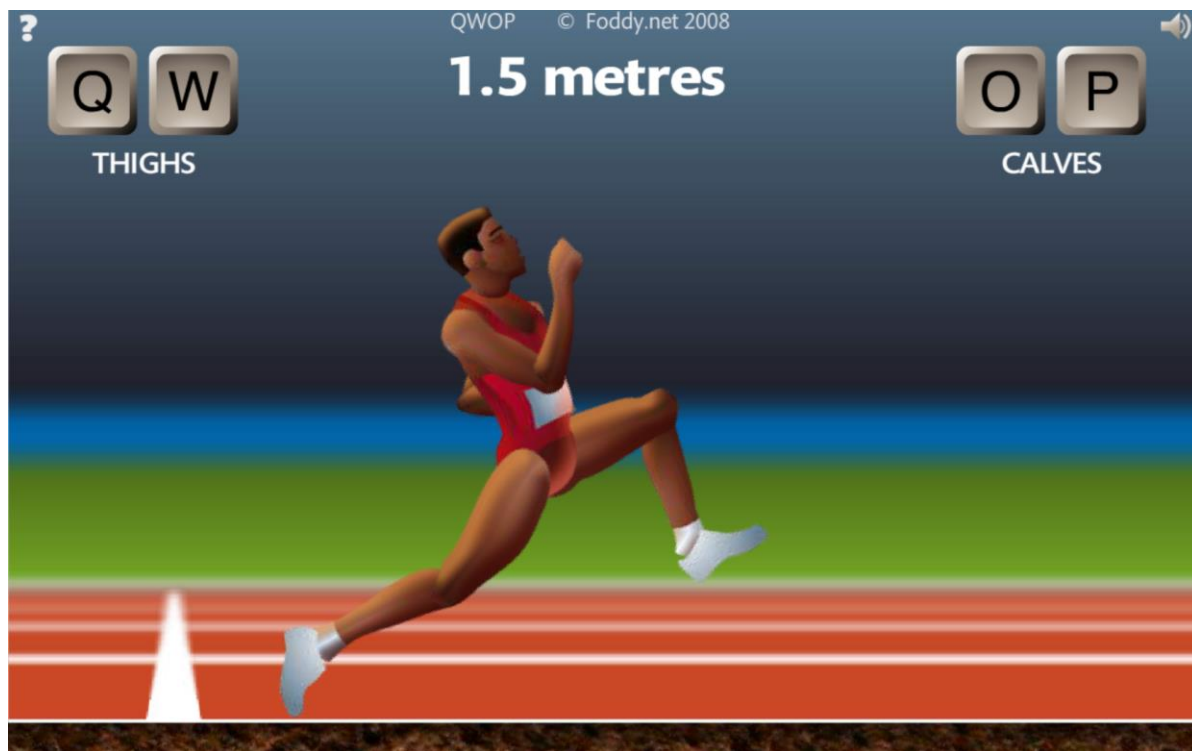
# 순서

1 강화학습 개념

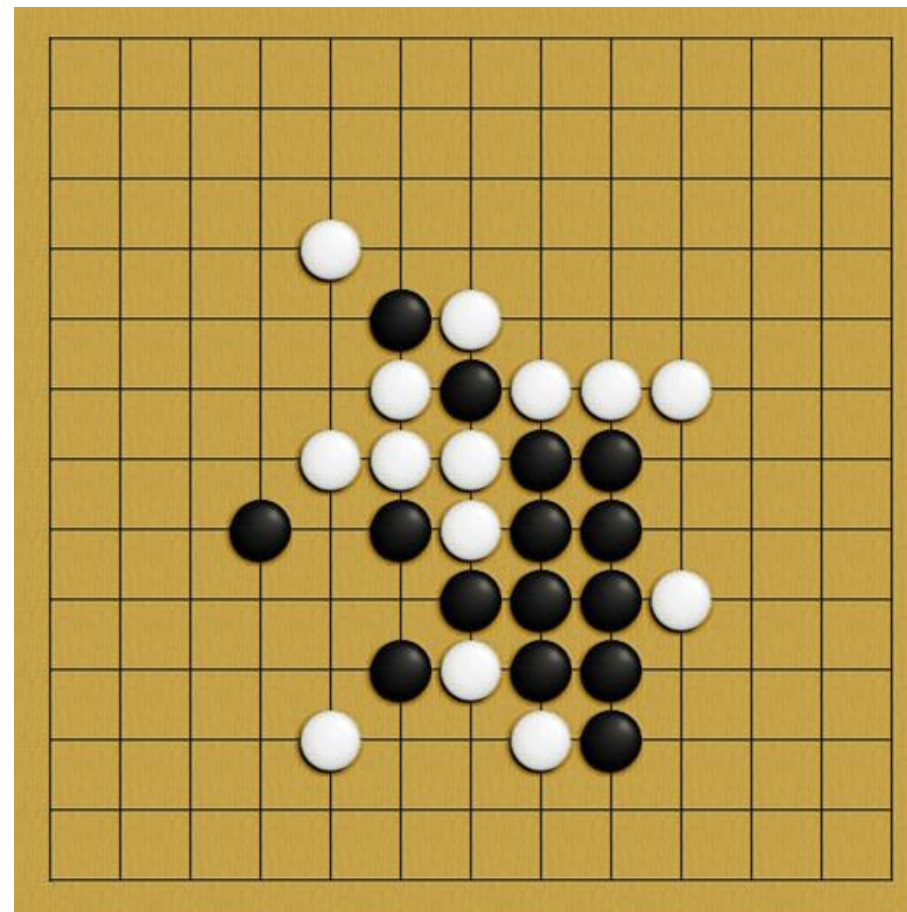
2 QWOP 달리기 게임

3 오목



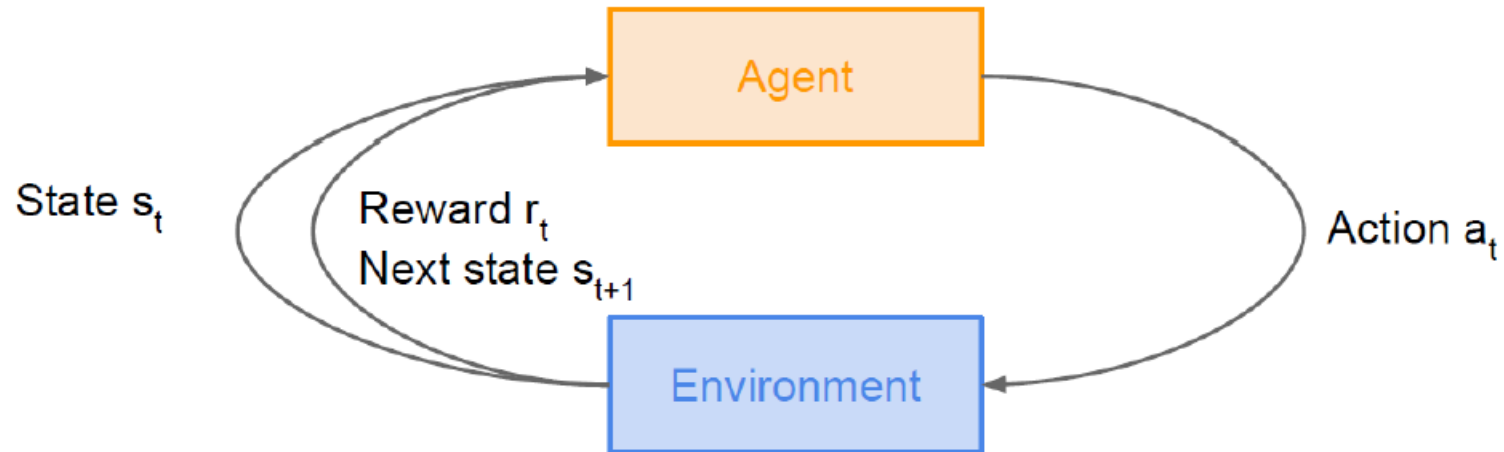


QWOP 달리기 게임



오목

# 강화학습 Reinforcement Learning



- Goal : Learn how to take **actions** to maximize **reward**
- An agent is interacting with an environment by following reward

# 마르코프 결정 프로세스 Markov Decision Process

- 마르코프 결정 프로세스 (MDP)
  - 순차적 행동 결정 문제를 수학적으로 정의한 것
    - 상태, 행동, 보상, 상태변환확률, 감가율, 정책으로 구성

$S$  : set of possible **state**

$A$  : set of possible **actions**

$R$  : **distribution of reward** given state and action

$P$  : transition probability; distribution over next state given state and action

$\gamma$  : discount factor

# 마르코프 결정 프로세스 Markov Decision Process

- 마르코프 결정 프로세스 (MDP)
  - 벨만 방정식이 성립하려면 학습 대상이 MDP이어야 함
  - 다음 단계의 상태  $\mathbf{s}_{t+1}$ 이 현재 상태  $\mathbf{s}_t$ 에서 취한 행동  $\mathbf{a}_t$ 에 의해 결정되는 시스템
- MDP가 아닌 것
  - 현재 상태  $\mathbf{s}_t$  외의 과거, 예를 들면  $\mathbf{s}_{t+1}$ 이  $\mathbf{s}_{t-1}$ 로부터도 영향을 받는 시스템

# 보상 Reward

- 미로 : 목표에 도달했을 때
- 로봇 : 넘어지지 않고 걸어난 거리
- 바둑 : 대국의 승리





# 보상 Reward

- 미로 : 목표에 도달했을 때
- 로봇 : 넘어지지 않고 걸어난 거리
- 바둑 : 대국의 승리



**QWOP와 오목의 보상체계**

# 보상 Reward

- **$R_t$** : 어떤 시각  $t$ 에 받을 수 있는 즉각 보상
  - 강화학습에서는 보상  **$R_t$** 를 태스크에 맞게 적절하게 결정해야
- **$G_t$** : 앞으로 받을 수 있으리라 예상되는 총 보상
  - 이 경우 시간의 경과에 따른 감가율을 고려해야 함

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \dots$$

# 가치함수 value function

- 상태가치함수 state value function

$$V^{\pi}(s) = E \left[ \sum_{t \geq 0} \gamma^t r_t | s, \pi \right]$$

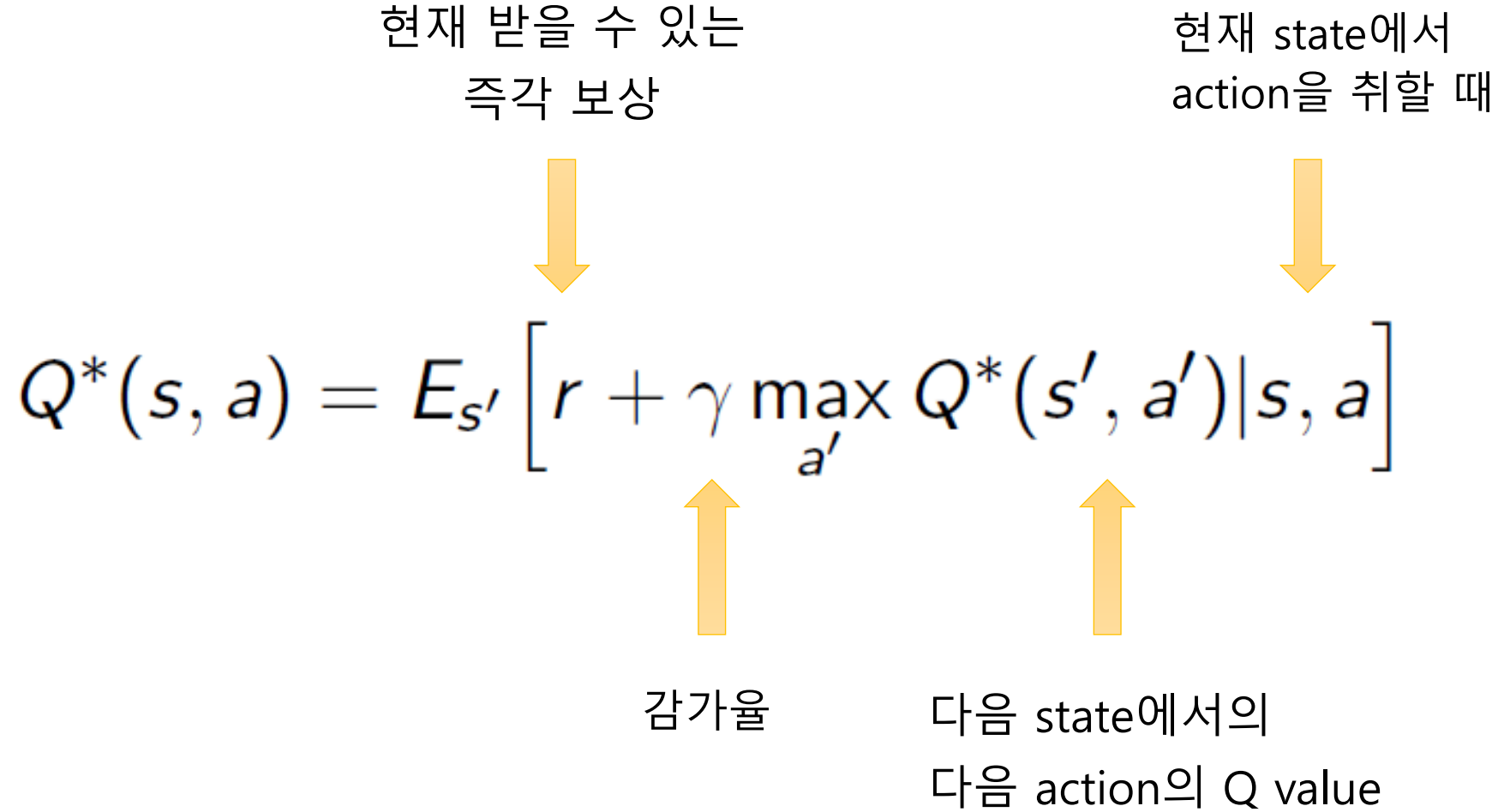
- 행동가치함수 action value function

$$Q^{\pi}(s, a) = E \left[ \sum_{t \geq 0} \gamma^t r_t | s, a, \pi \right]$$

# 벨만방정식 Bellman equation

현재 받을 수 있는  
즉각 보상

현재 state에서  
action을 취할 때



The diagram illustrates the Bellman equation with four orange arrows pointing to its components: one from the reward term  $r$  (labeled '현재 받을 수 있는 즉각 보상'), one from the discount factor  $\gamma$  (labeled '감가율'), one from the next state  $s'$  (labeled '다음 state에서의'), and one from the next action  $a'$  (labeled '다음 action의 Q value').

$$Q^*(s, a) = E_{s'} \left[ r + \gamma \max_{a'} Q^*(s', a') | s, a \right]$$

감가율

다음 state에서의  
다음 action의 Q value

# 벨만방정식 Bellman equation

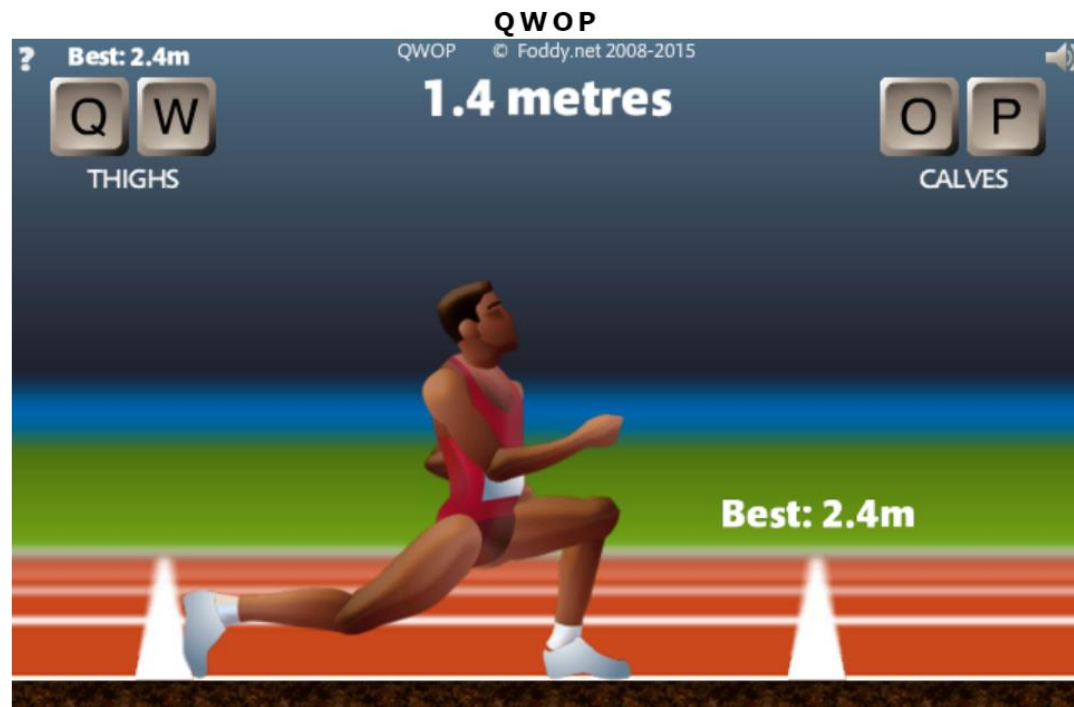
$$Q^*(s, a) = E_{s'} \left[ r + \gamma \max_{a'} Q^*(s', a') | s, a \right]$$



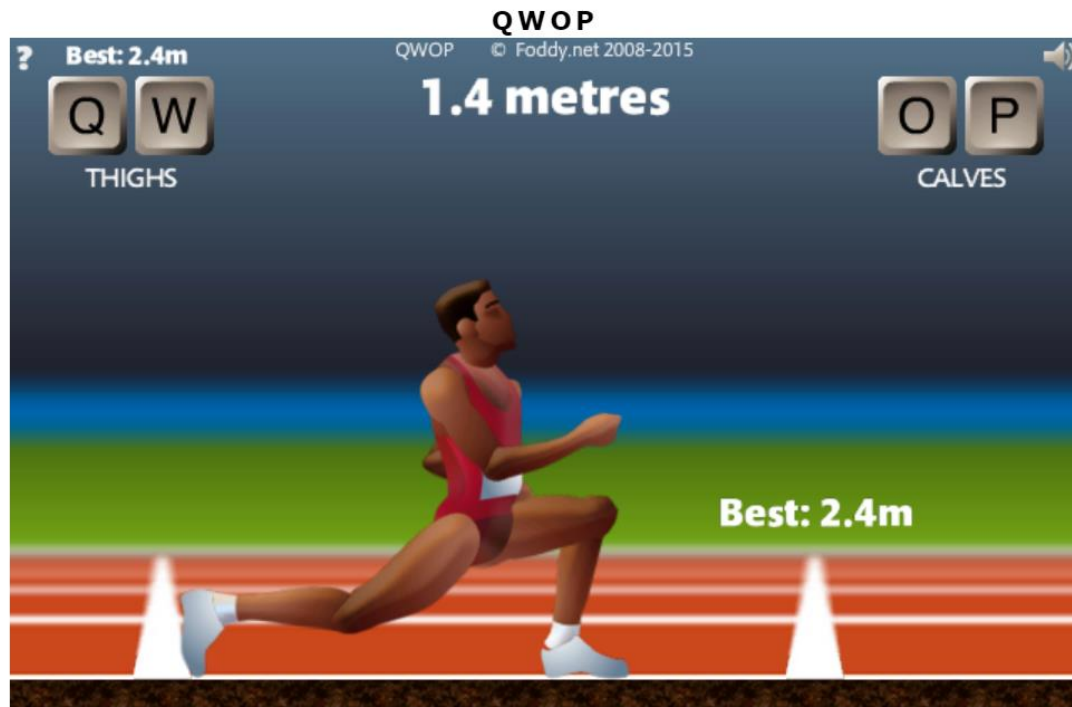
Q를 최대화하는  
action을 찾자!

QWOP Game

# QWOP GAME



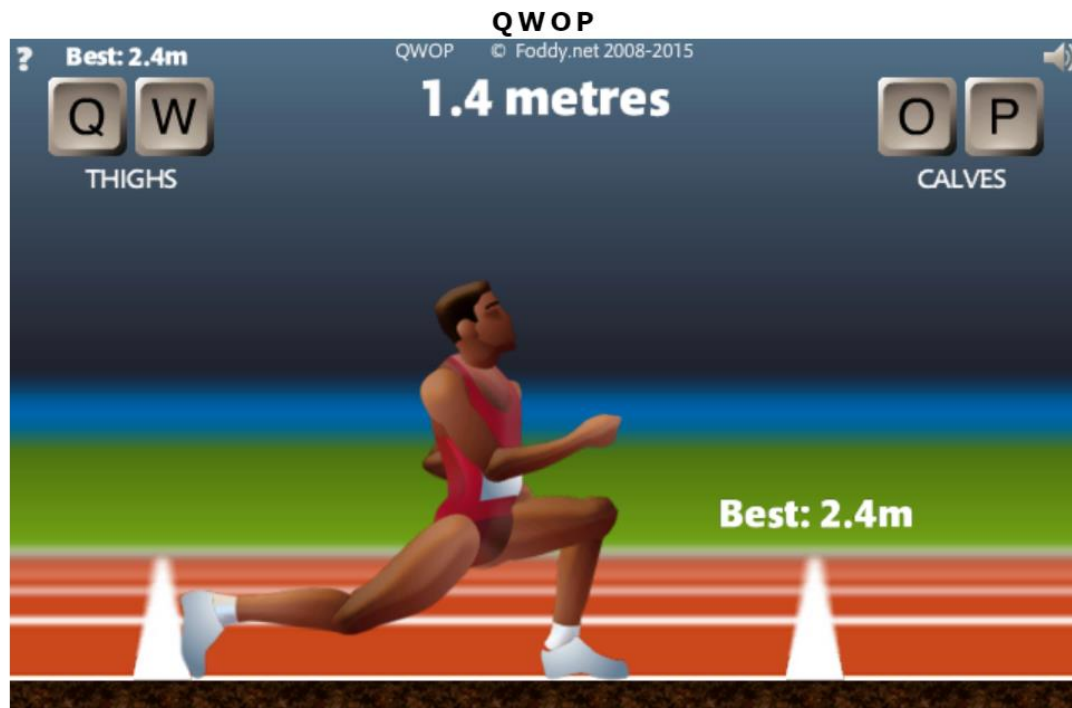
# QWOP GAME



Q W O P 4Key만을 눌러서  
최대한 멀리 달리는 인터넷에서  
실행하는 플래시 게임입니다



# QWOP GAME



Q W O P 4Key만을 눌러서  
최대한 멀리 달리는 인터넷에서  
실행하는 플래시 게임입니다

## 진짜 어렵습니다

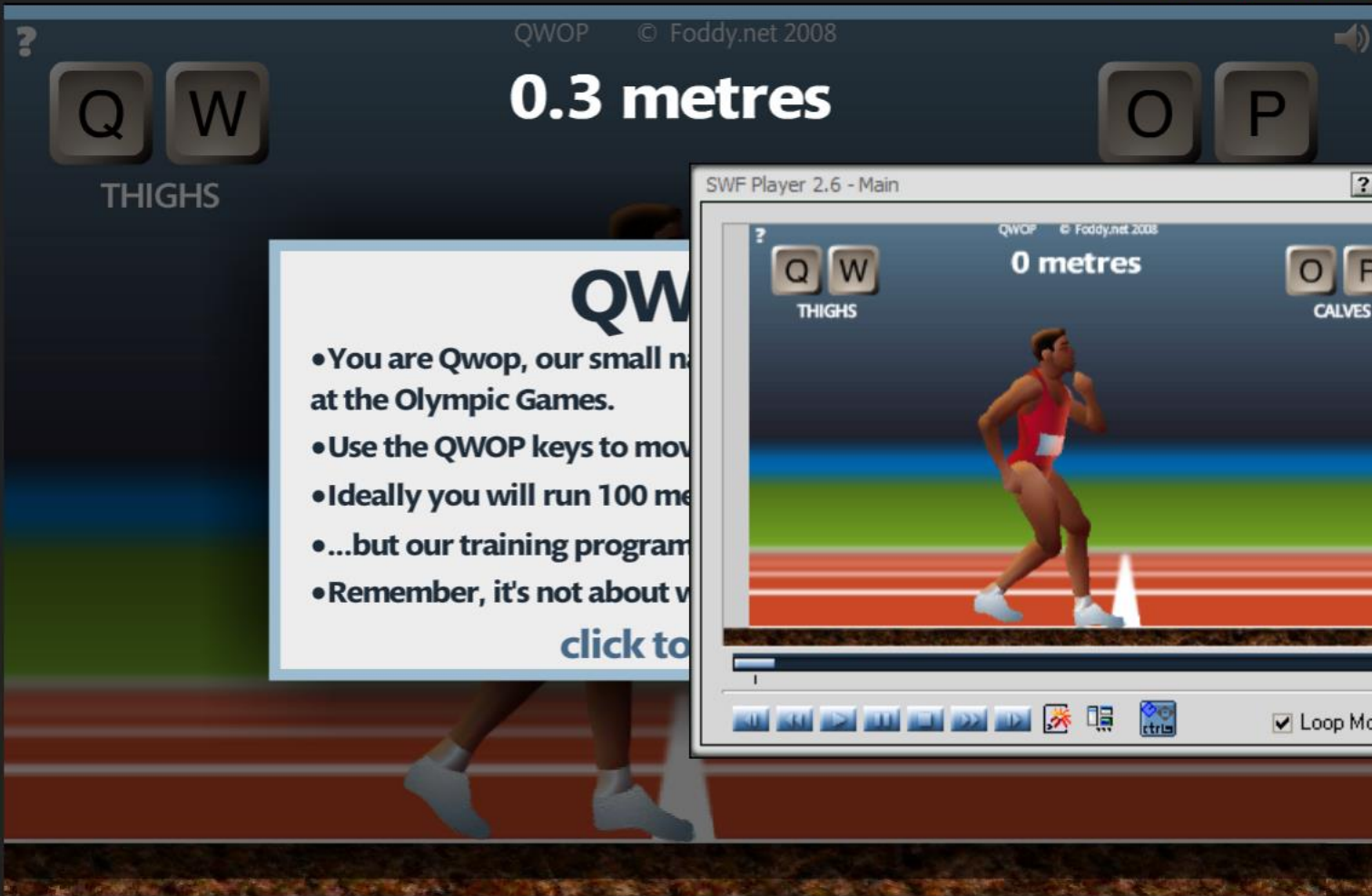
많이 어려워요 ㅠㅠ



# 내 컴퓨터에서 플레이하기



**SWF Player** for Windows



**QWOP**

- You are Qwop, our small n...
- Use the QWOP keys to mov...
- Ideally you will run 100 me...
- ...but our training program...
- Remember, it's not about v...

click to



Elements Console Sources

```

<div id="page_wrap">
  <div class="col-md-12 header">...</div> == $0
  <div class="col-md-12 box-game">
    <div class="col-md-12" style="height: auto !important; min-height: 0px !important;">...</div>
    <div class="col-md-2 box-left" style="width: 792px;">...</div>
    <div class="col-md-8 box-center" style="width: 721px;">
      <div class="box-title">...</div>
      <div class="box-play" data-width="640" data-height="400" style="width: 757px; height: 491.875px;">
        <object type="application/x-shockwave-flash" data="http://game.qwop.online/swf/qwop.swf" width="100%" height="100%" id="flashgame">...</object>
      </div>
    </div>
  </div>
</div>
  
```

body div#page\_wrap div.col-md-12.header

1 of 2 Cancel

Files Event Listeners DOM Breakpoints Properties Accessibility

er :hov .cls +

ment.style {

position 0

margin -

border -

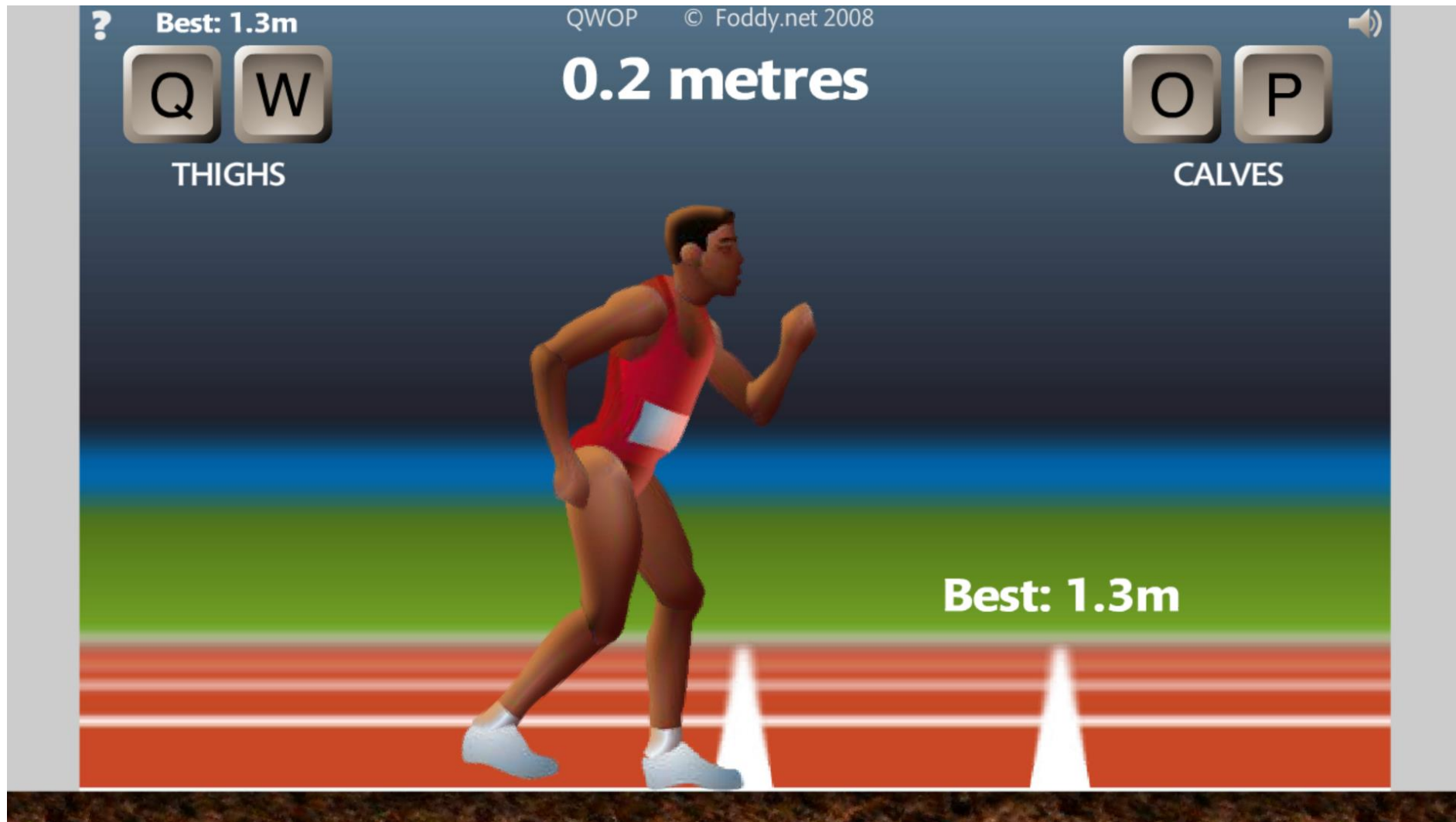
Console What's New

해보실 분~!

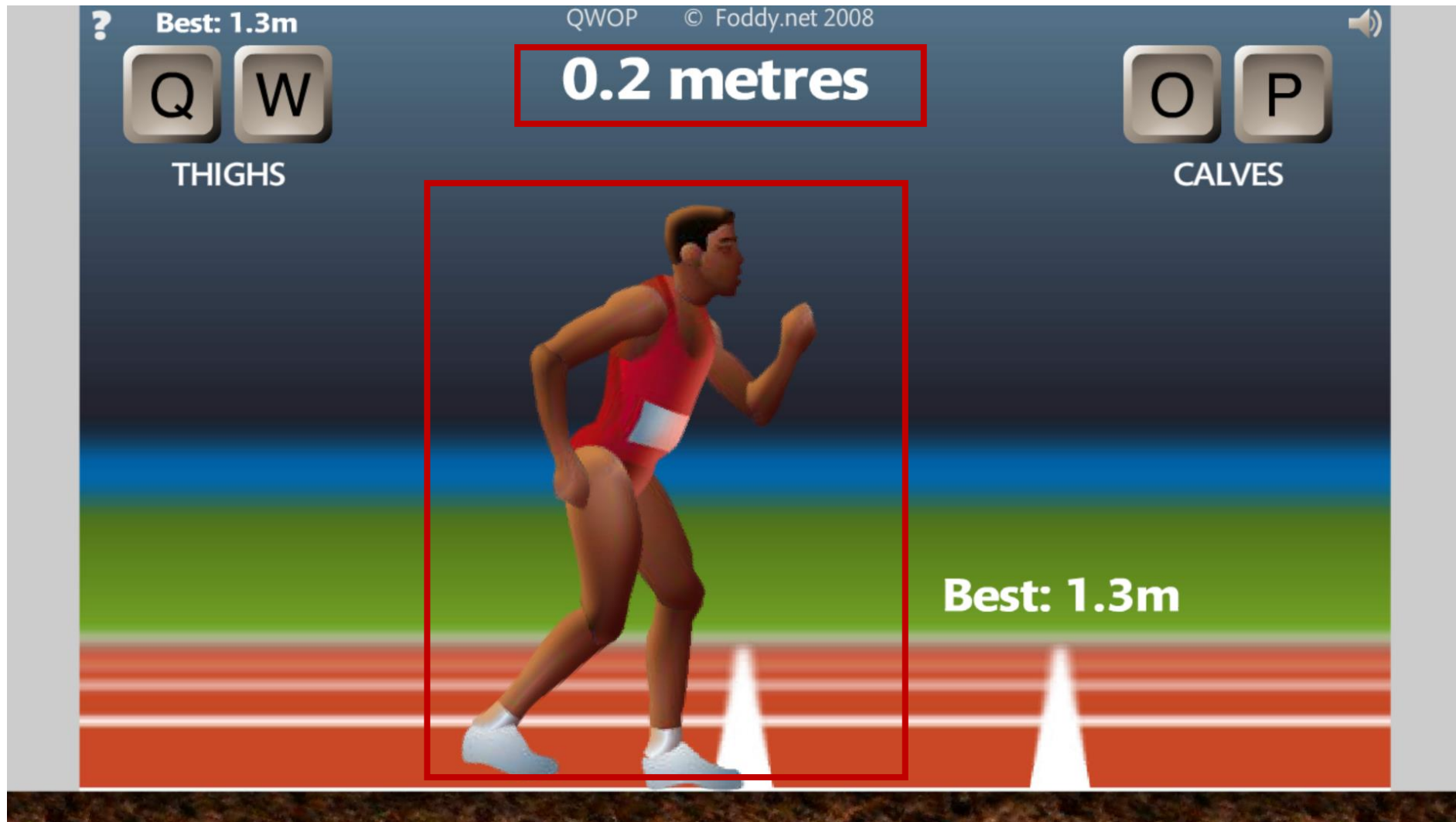
# 해보실 분~!

아무도 손 안 들 줄 알았어요  
회장님 나와주세요~

# 필요한 정보

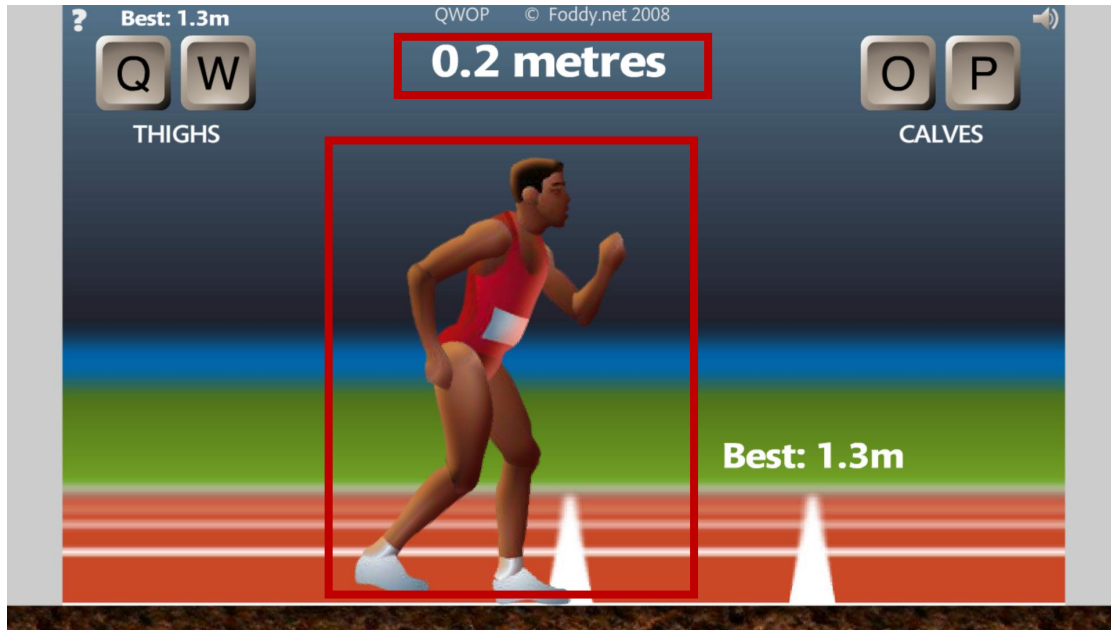


# 필요한 정보





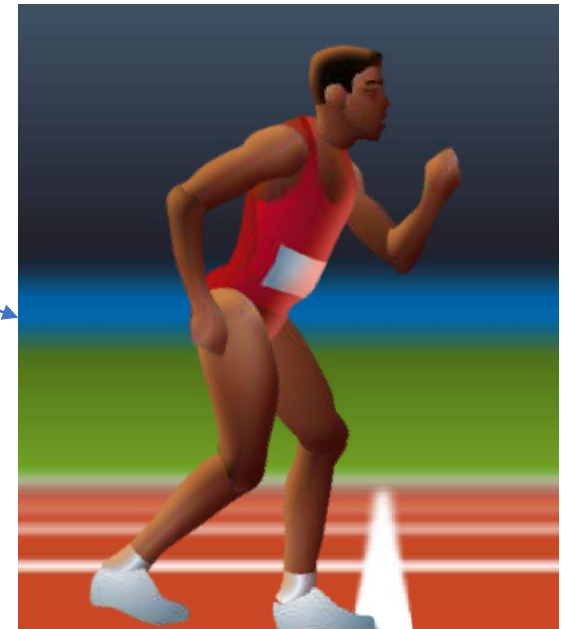
# PIL ImageGrab



**Reward**

**0.2 metres**

**State**





**0.2 metres**



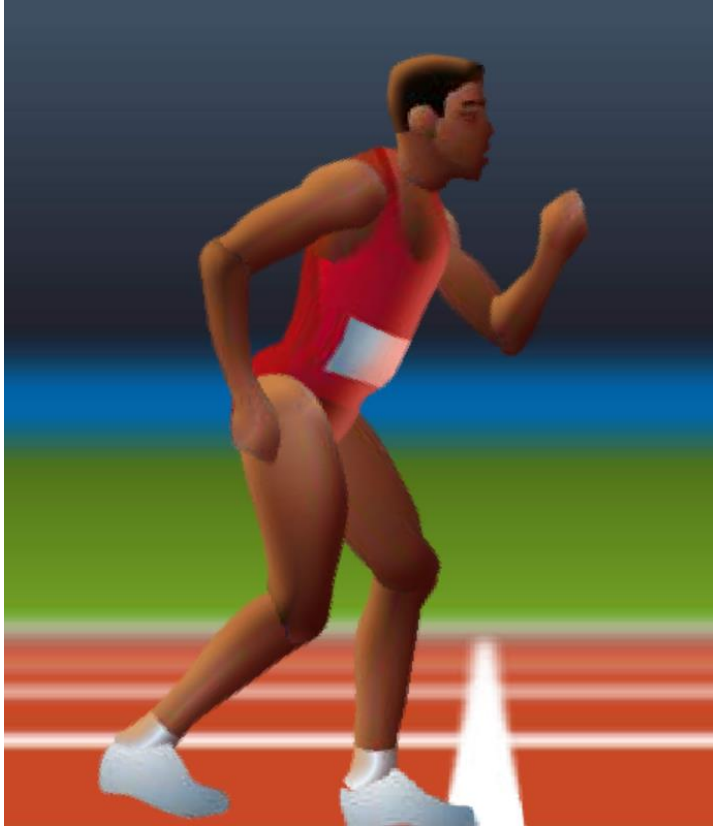
float  
0.2

**0.2 metres**

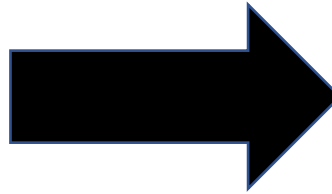


float  
0.2

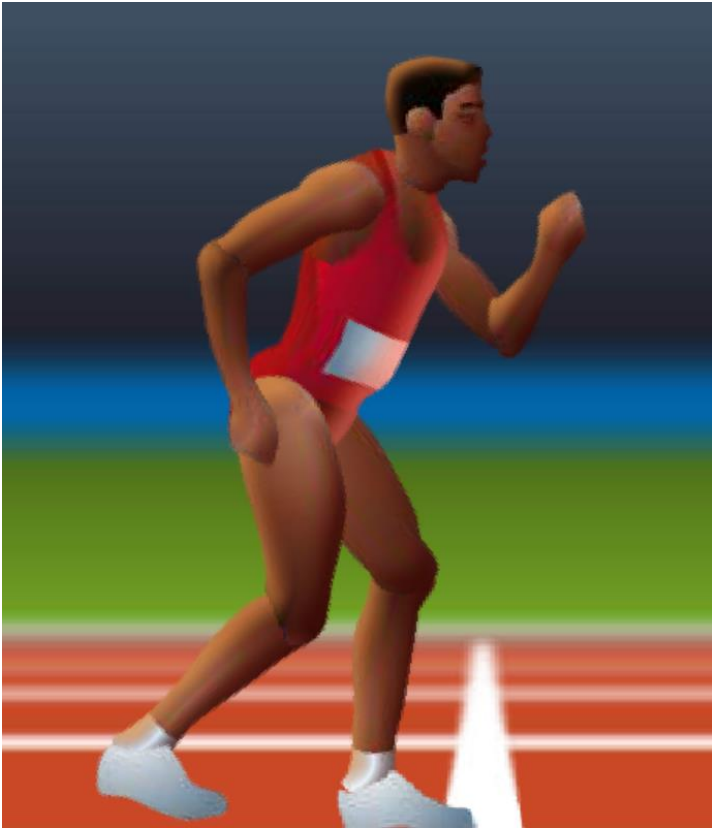
850 x 900



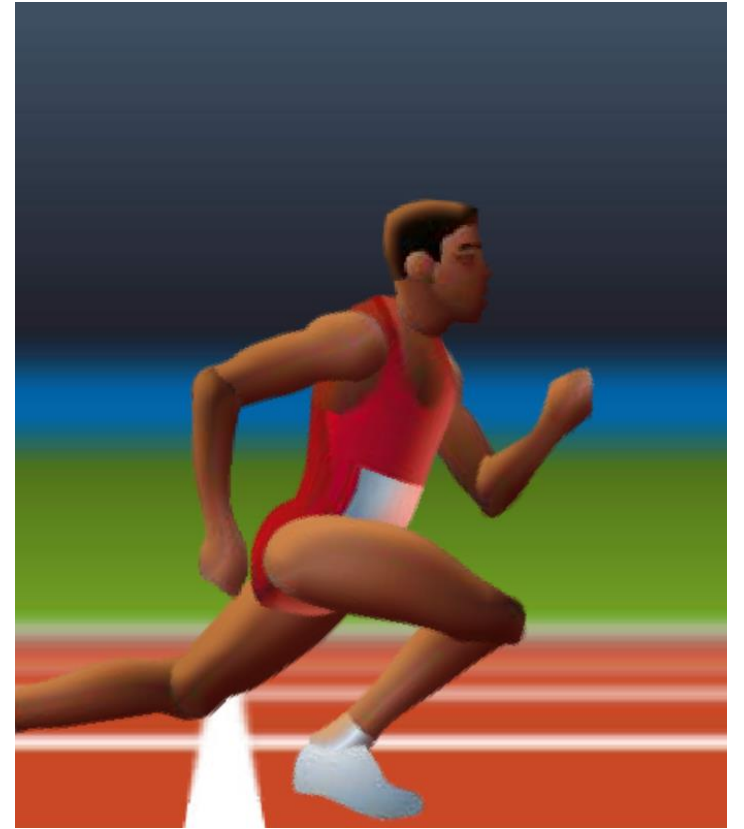
65 x 80 numpy array



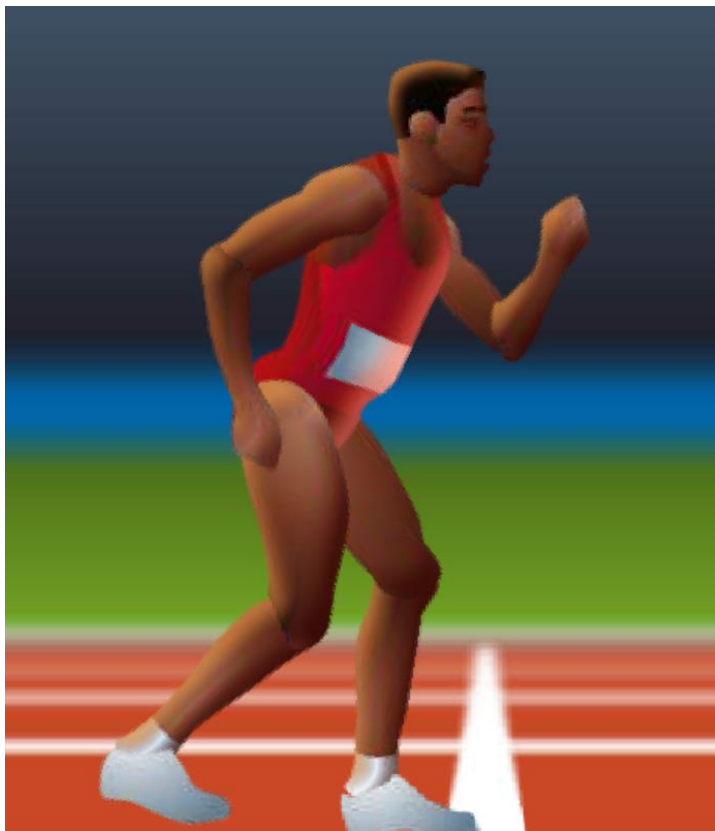
```
array([[0.32542693, 0.71325745, 0.13087782, ..., 0.78405749, 0.93163142,  
        0.10993886],  
       [0.35862853, 0.60352696, 0.55624166, ..., 0.69840159, 0.70768583,  
        0.09113853],  
       [0.8143676 , 0.66985932, 0.38867219, ..., 0.46066248, 0.205031 ,  
        0.50635868],  
       ...,  
       [0.07542664, 0.50524852, 0.81975087, ..., 0.14056735, 0.89529886,  
        0.74117241],  
       [0.76673731, 0.25676113, 0.62633748, ..., 0.16967623, 0.16171968,  
        0.2455536 ],  
       [0.86042703, 0.59356211, 0.76426715, ..., 0.79938846, 0.31258342,  
        0.97210255]])
```



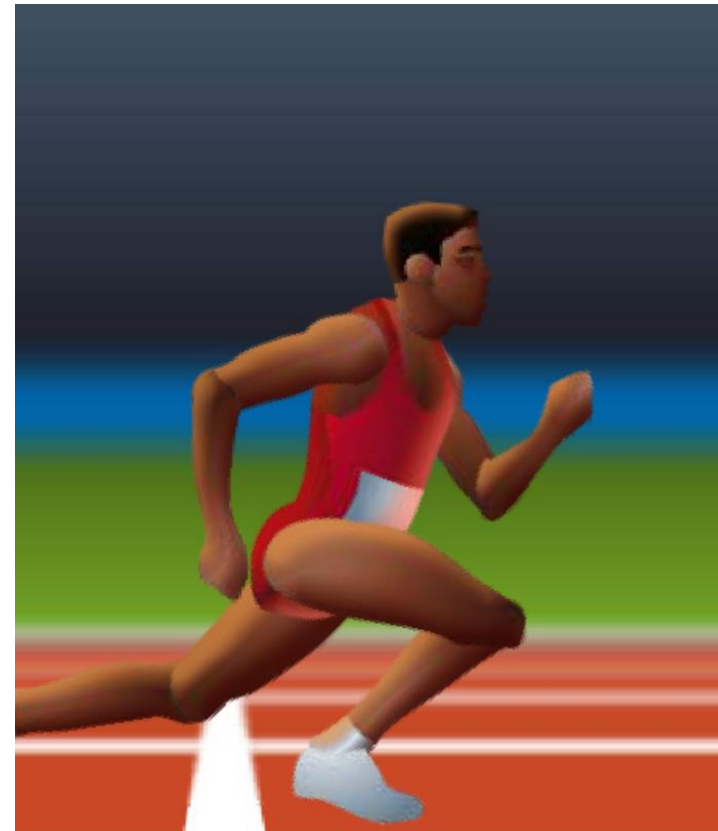
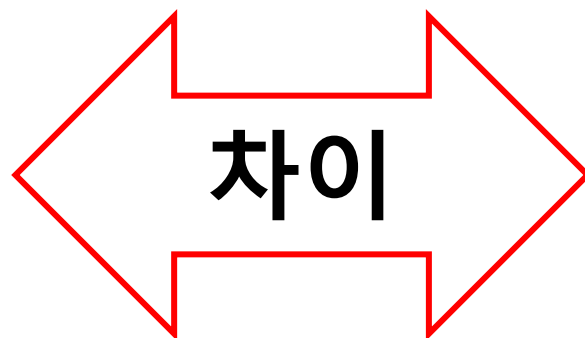
이전 상태



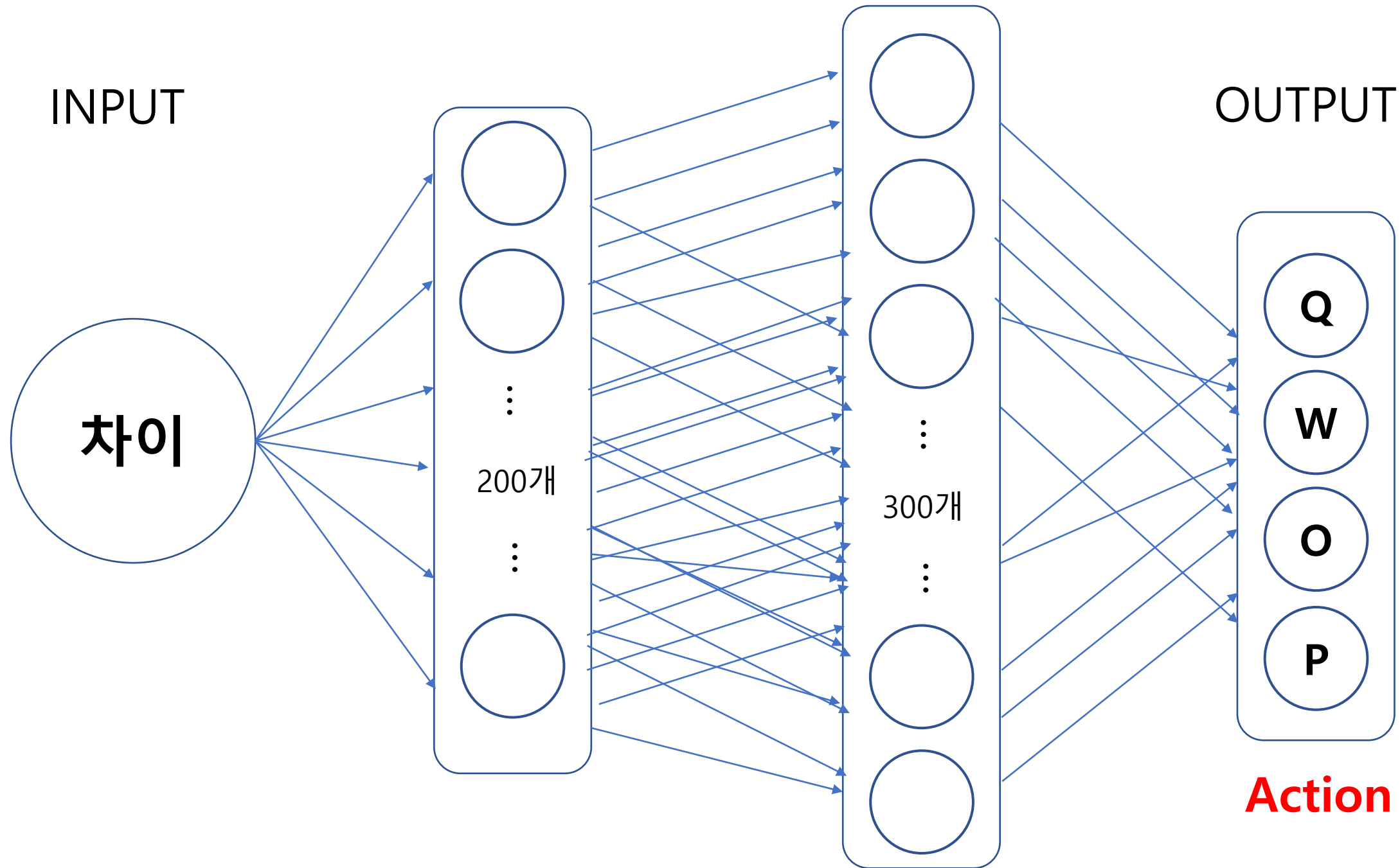
현재 상태



이전 상태



현재 상태



# Output

명령어 (q w o p) 중에 어떤 걸 선택하는게  
좋은 결과를 내는지에 대한 확률

while True:

for-loop 50번:

게임실행

게임 끝나면 정보 저장

50번 동안의 게임 정보를 토대로  
네트워크 학습



while True:

for-loop 50번:

게임실행

게임 끝나면 정보 저장

50번 동안의 게임 정보를 토대로  
네트워크 학습

Reward

얼마나 멀리 갔는지

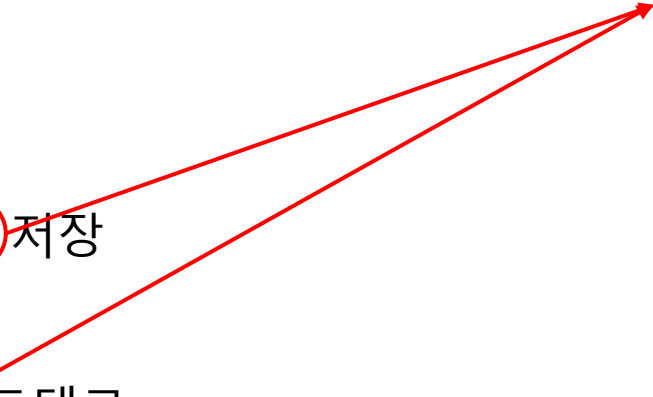
얼마나 많은 시간이 걸렸는지

Probability

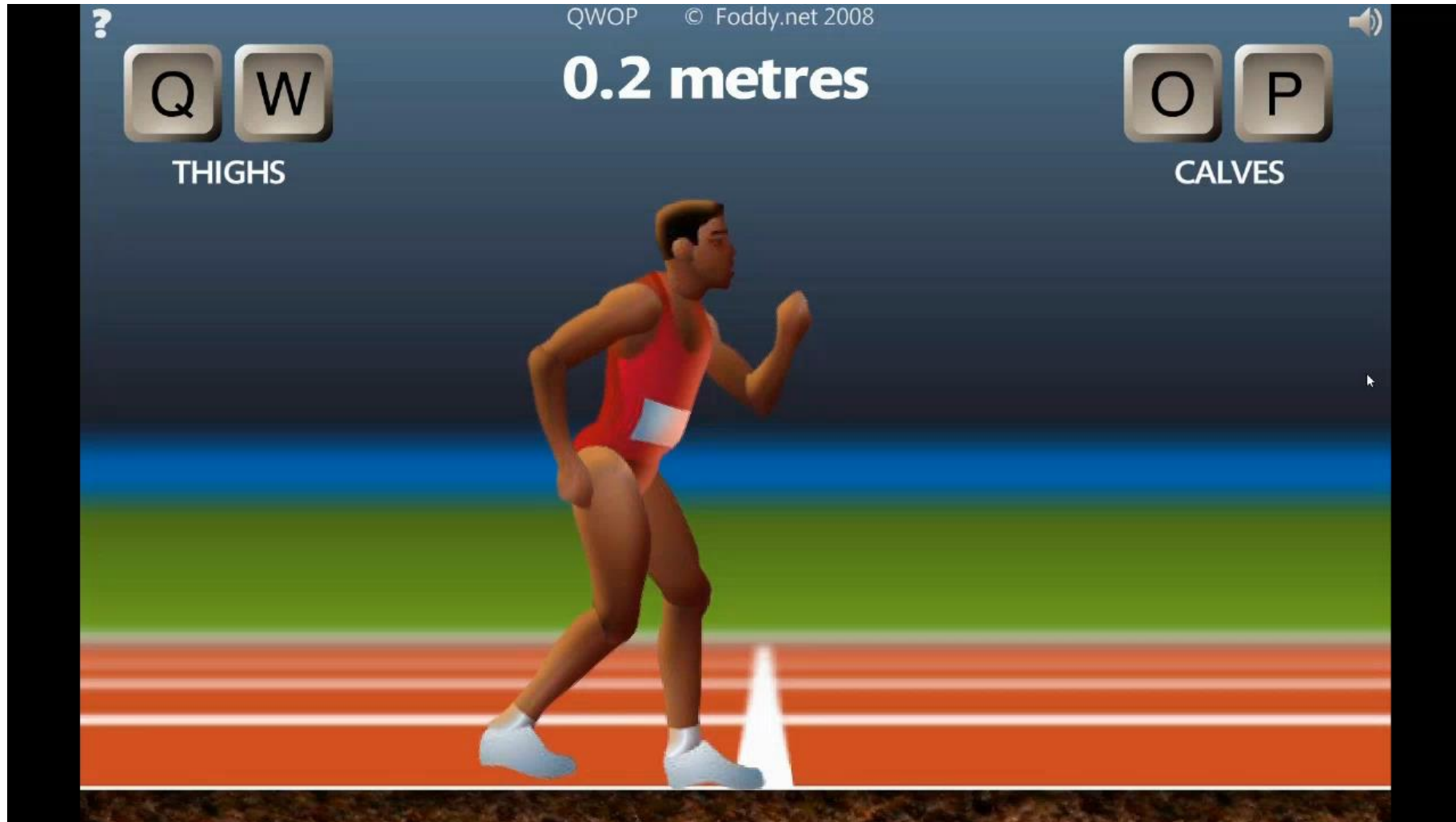
어떤 확률 분포를 가질 때  
가장 좋은 결과가 나오는지

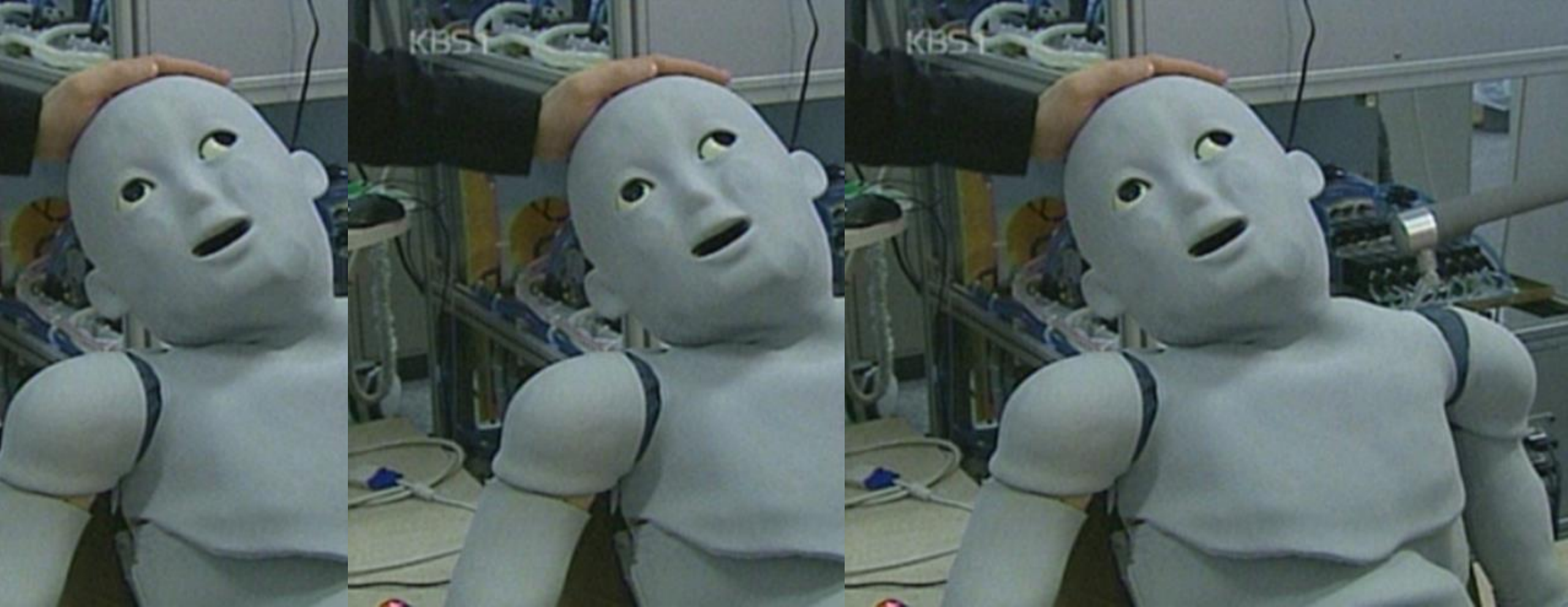
정보

정보



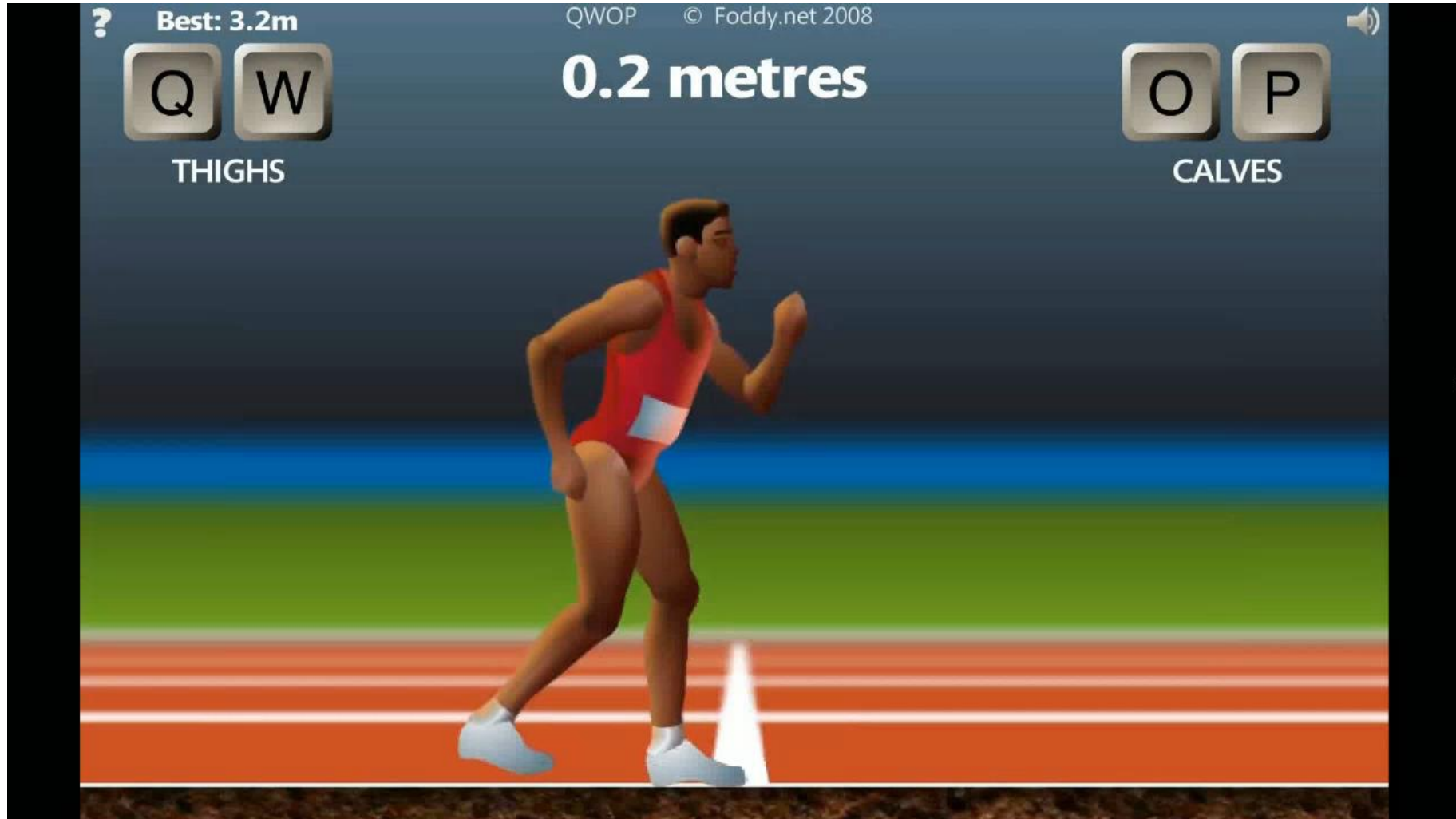
# 학습 초기 모델





학습 중...

# 학습 후 모델



오목

표

**Actor-Critic**

**MCTS**

**Replay buffer**

# Actor-Critic

“

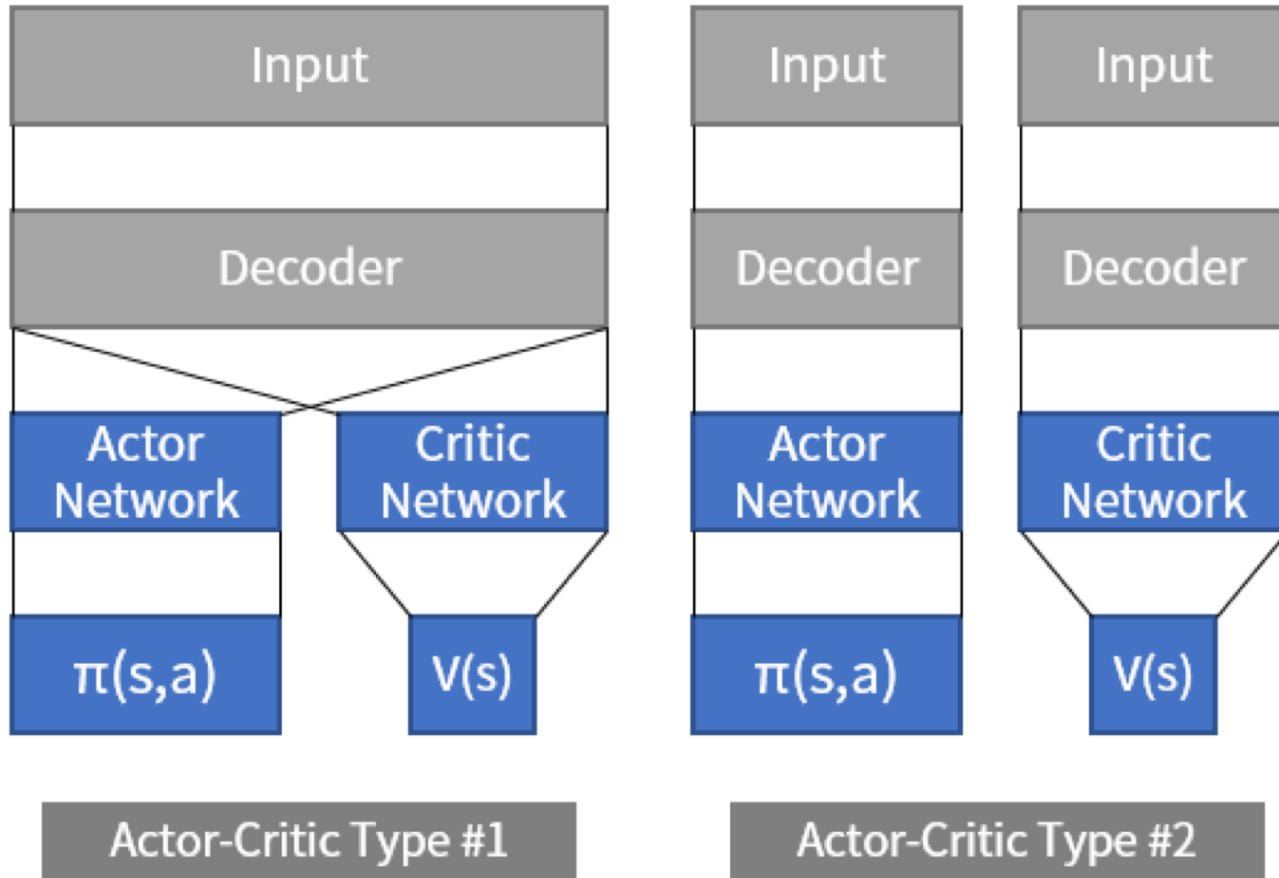
*Actor : 상태가 주어졌을 때 행동을 결정*

*Critic : 상태의 가치를 평가*

*이 두개의 network를 사용*



# Actor Critic



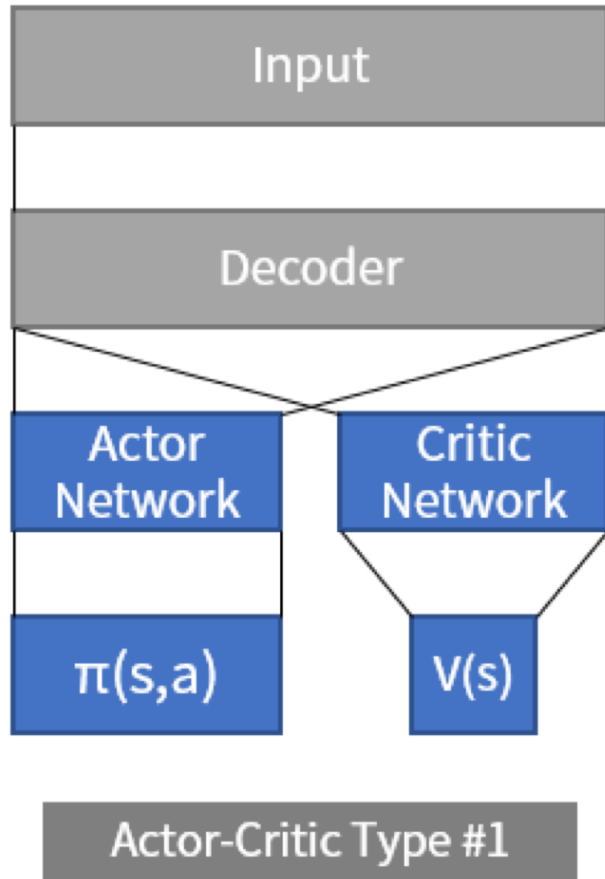
## Actor

- 어떠한 action을 선택할지 결정하는 역할
- state와 action을 input으로 받아 결정

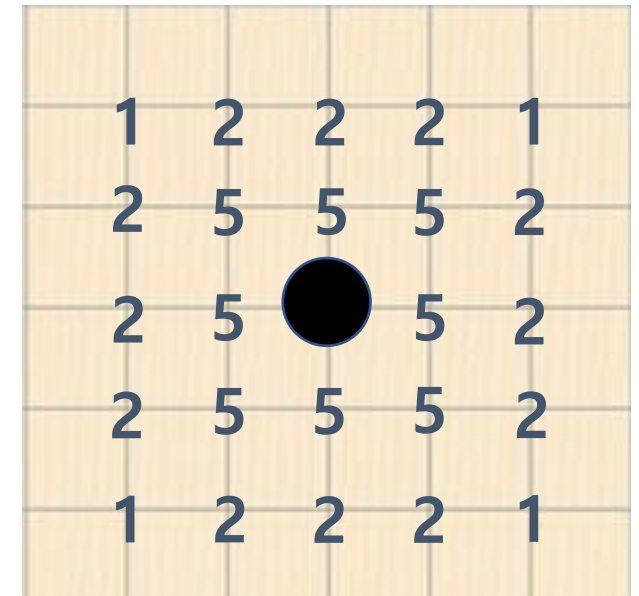
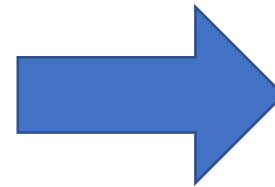
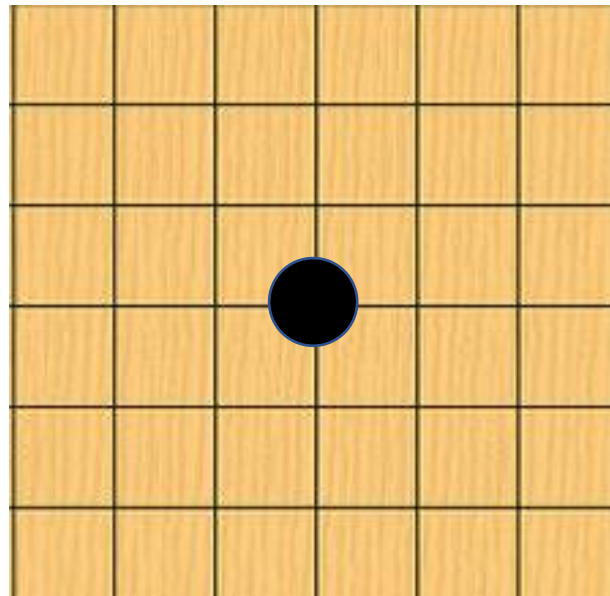
## Critic

- 현재 State에 대한 가치함수
- state를 input으로 받아 가치 평가

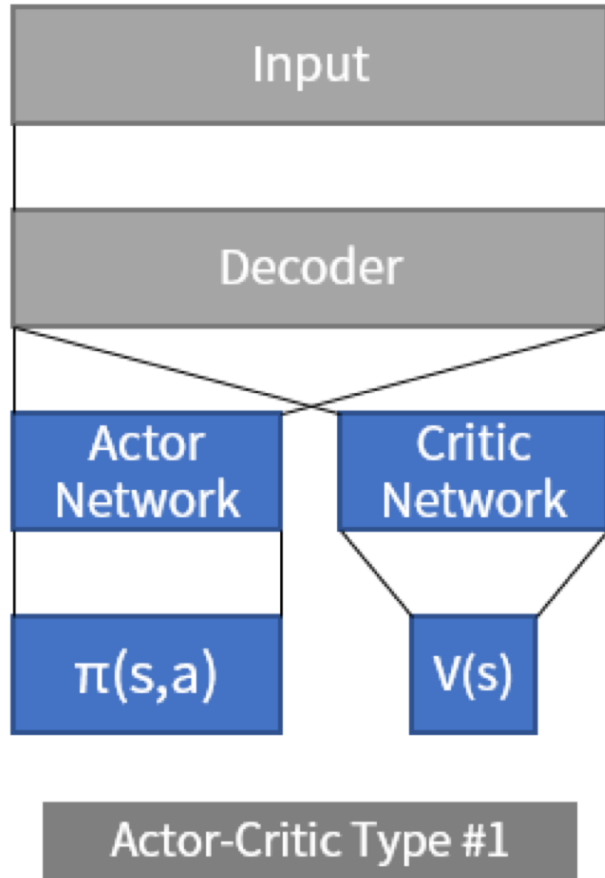
# Actor Critic



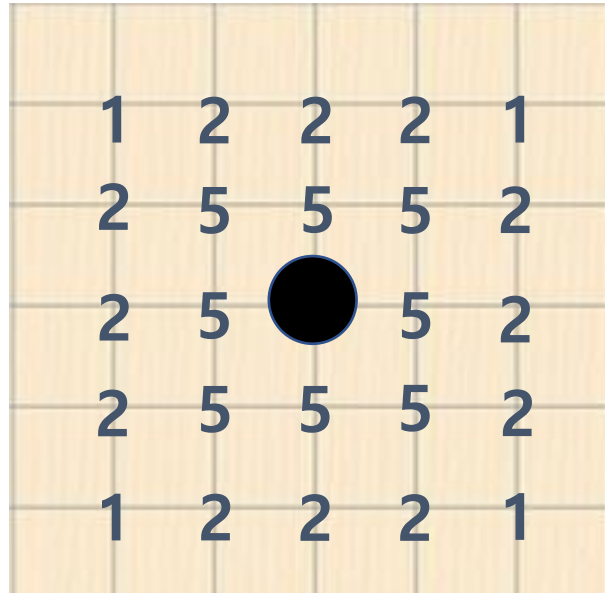
추가 사항



# Actor Critic



**Actor** : 어떠한 action을 선택할지 결정하는 역할



```
Adj_board += pi
adj_board /= adj_board.sum( )
action = np.random.choice(action_size,
                           p=adj_board)
```

```
# np.random.choice(a,p)
# n = data , p = percentage
```

```
# pi = visit / visit.sum()
```

# M C T S

Monte-Carlo Tree Search



AlphaGo

---

# MCTS

## 1. 조건

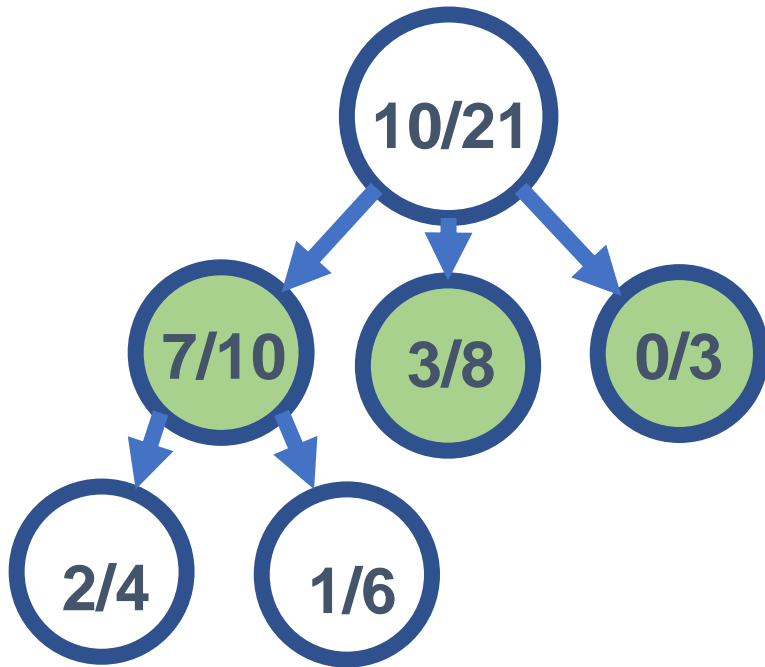
- 1) 최대, 최소 점수 값이 존재
- 2) 게임 규칙 존재 & 완전 정보
- 3) 게임 길이 제한

## 2. 순서

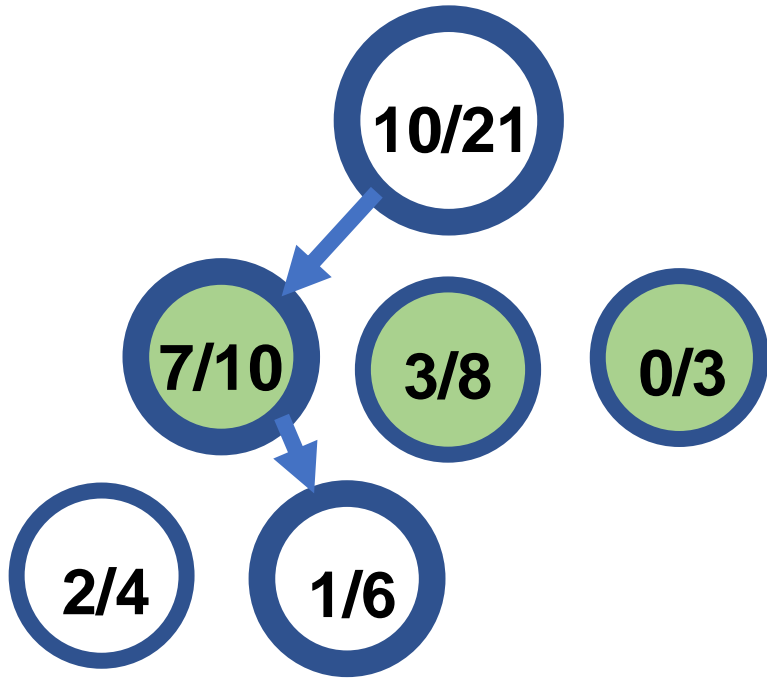
- 1) 선택 ( Selection )
- 2) 확장 ( Expansion )
- 3) 시뮬레이션 ( Simulation )
- 4) 역전파 ( Backpropagation )

# MCTS

0. Node는 reward 와 visit으로 구성



# MCTS - Selection

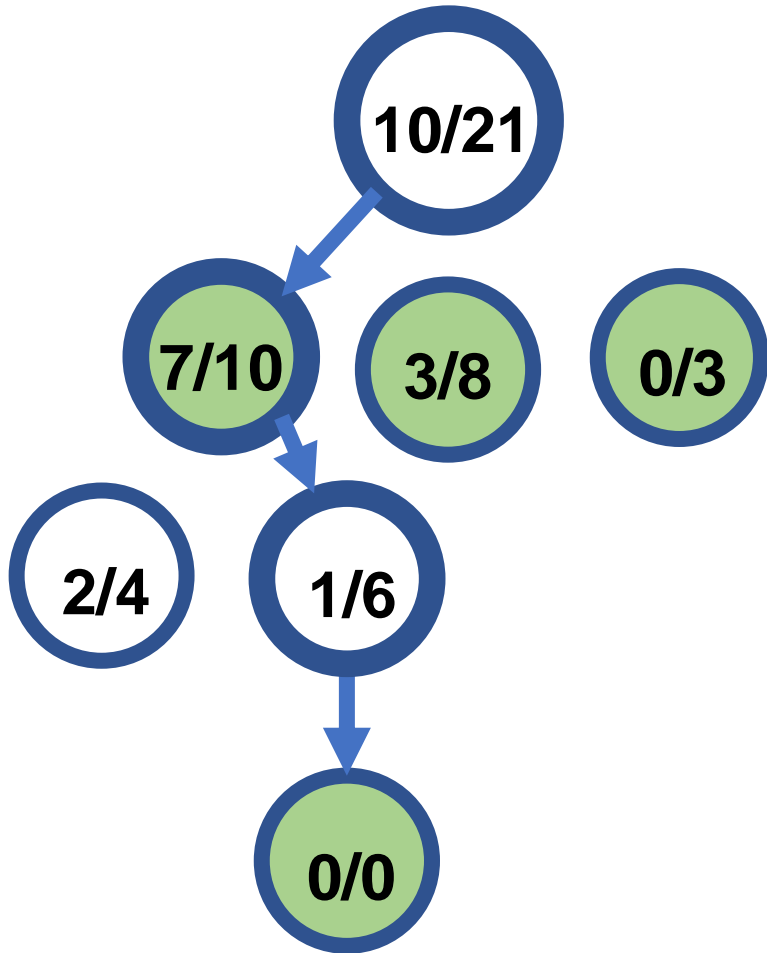


0. Node는 reward 와 visit으로 구성

1. 갈 수 있는 Node중에서 visit 대비 reward가 높은 Node로 이동



# MCTS - Expansion

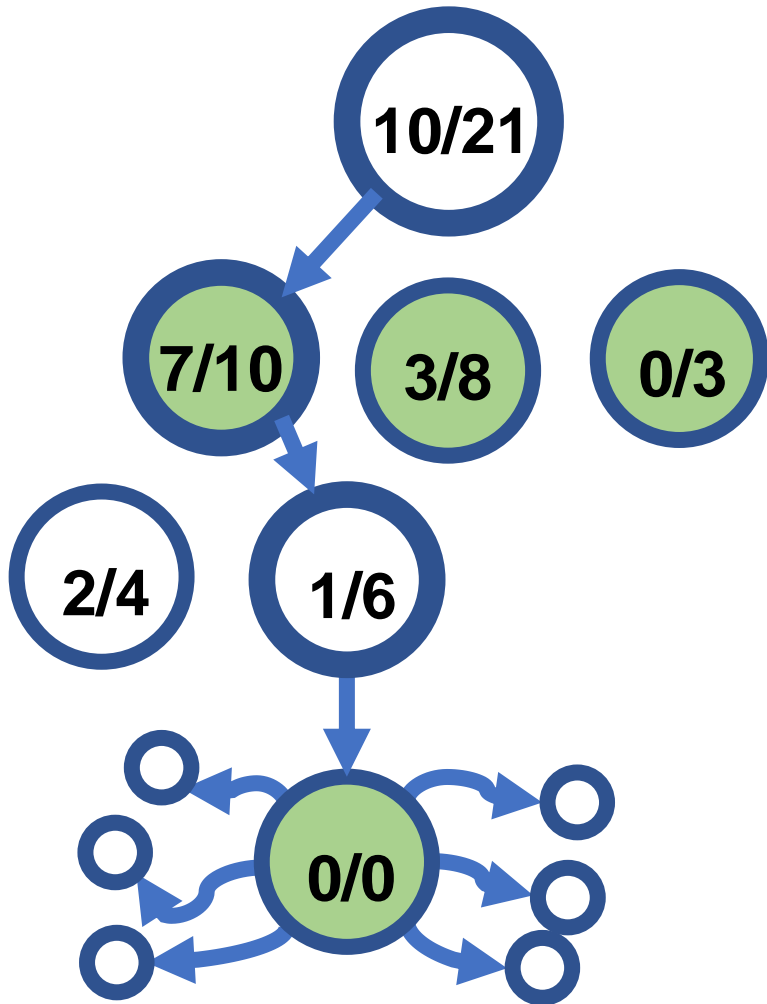


0. Node는 reward 와 visit으로 구성

1. 갈 수 있는 Node중에서 visit 대비 reward가 높은 Node로 이동

2. 마지막 Node에서 게임이 종료되지 않는다면, 트리에 존재하지 않은 다음 턴을 선택하여 노드 추가

# MCTS - Simulation



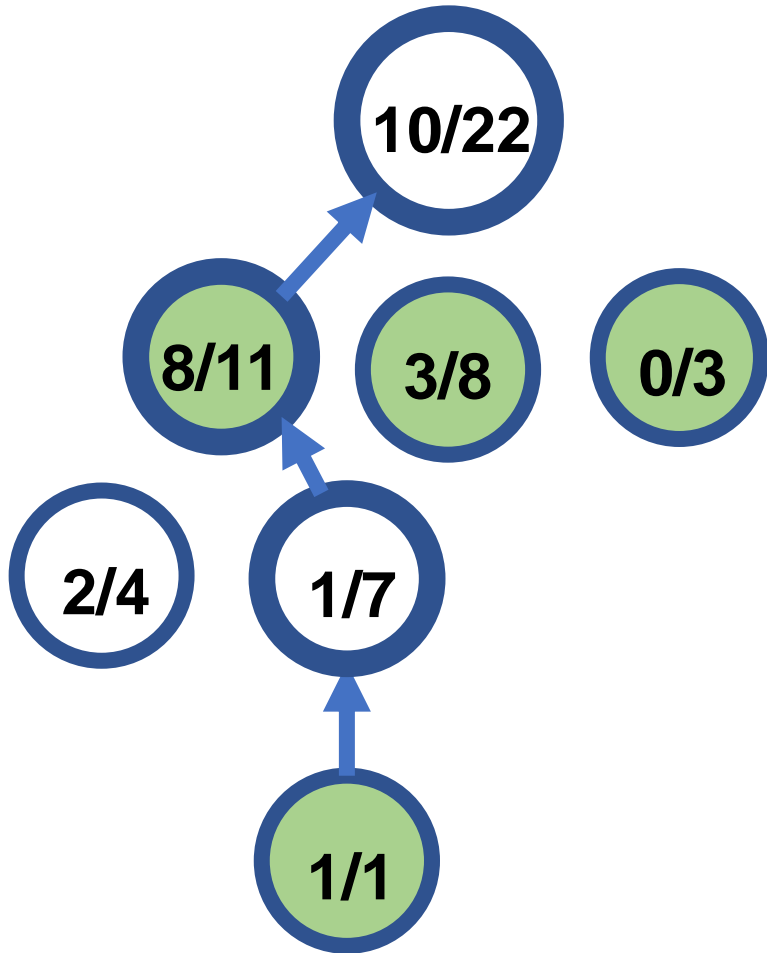
0. Node는 reward 와 visit으로 구성

1. 갈 수 있는 Node중에서 visit 대비 reward가 높은 Node로 이동

2. 마지막 Node에서 게임이 종료되지 않는다면, 트리에 존재하지 않은 다음 턴을 선택하여 노드 추가

3. 확장된 Node에서 랜덤으로 여러 번의 게임을 실행

# MCTS - Backpropagation



0. Node는 reward 와 visit으로 구성

1. 갈 수 있는 Node중에서 visit 대비 reward가 높은 Node로 이동

2. 마지막 Node에서 게임이 종료되지 않는다면, 트리에 존재하지 않은 다음 턴을 선택하여 노드 추가

3. 확장된 Node에서 랜덤으로 여러 번의 게임을 실행

4. 역전파를 통해 상단 노드에 게임 결과를 업데이트

# Replay Buffer

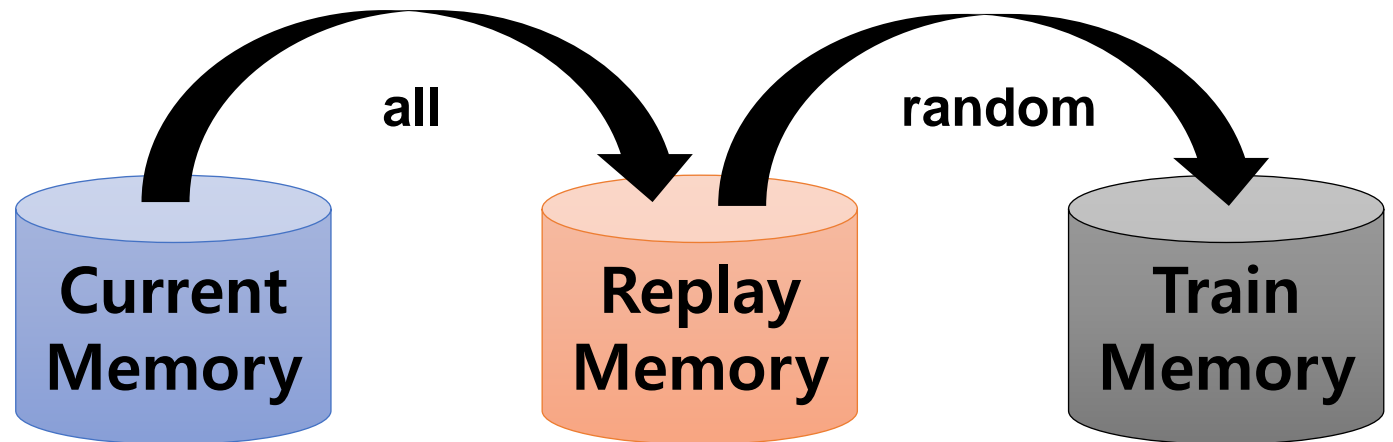
# MCTS – Replay Buffer

```
for epoch in range(epochs)
  For episode in range( Epi_count ):
    for i in range(n):
      MCTS( ) # 이때 current memory에 계속 data를 누적

    rep_buffer.extend(cur_buffer)
    cur_buffer.init()

  train_buffer.extend(random.sample(rep_buffer))

  Train(train_buffer)
```



# 한계점

두계점세계점네계점

만만할 줄 알았더니



1. 예상했던 것 보다 훨씬 높은 난이도
2. 좋은 장비를 보유하고 있지 않음
3. 2번의 이유로 생각했던 것보다 훨씬 학습이 오래 걸림.
4. 이러한 이유들로 인해 많은 시간이 필요했지만 충분한 시간을 확보하지 못함

그럼에도 불구하고  
해보실 분~!

그럼에도 불구하고  
해보실 분~!

**이 또 안계실꺼 같아서  
제가 해볼께요~~**