Use Case of Video Analytics in Intelligent Transport System

PREPARED BY: Dr Azhar Mohd Ibrahim

PhD Universiti Sains Malaysia (Collective Intelligence)

Embedded AI - DRL for Intelligent Robotics

PREPARED BY: Assoc Prof Dr Azhar Mohd Ibrahim

PhD Universiti Sains Malaysia (Collective Intelligence)

Coordinator of Advanced Multi-Agent System Lab,

Dept of Mechatronics Engineering,

International Islamic University Malaysia

Artificial General Intelligence

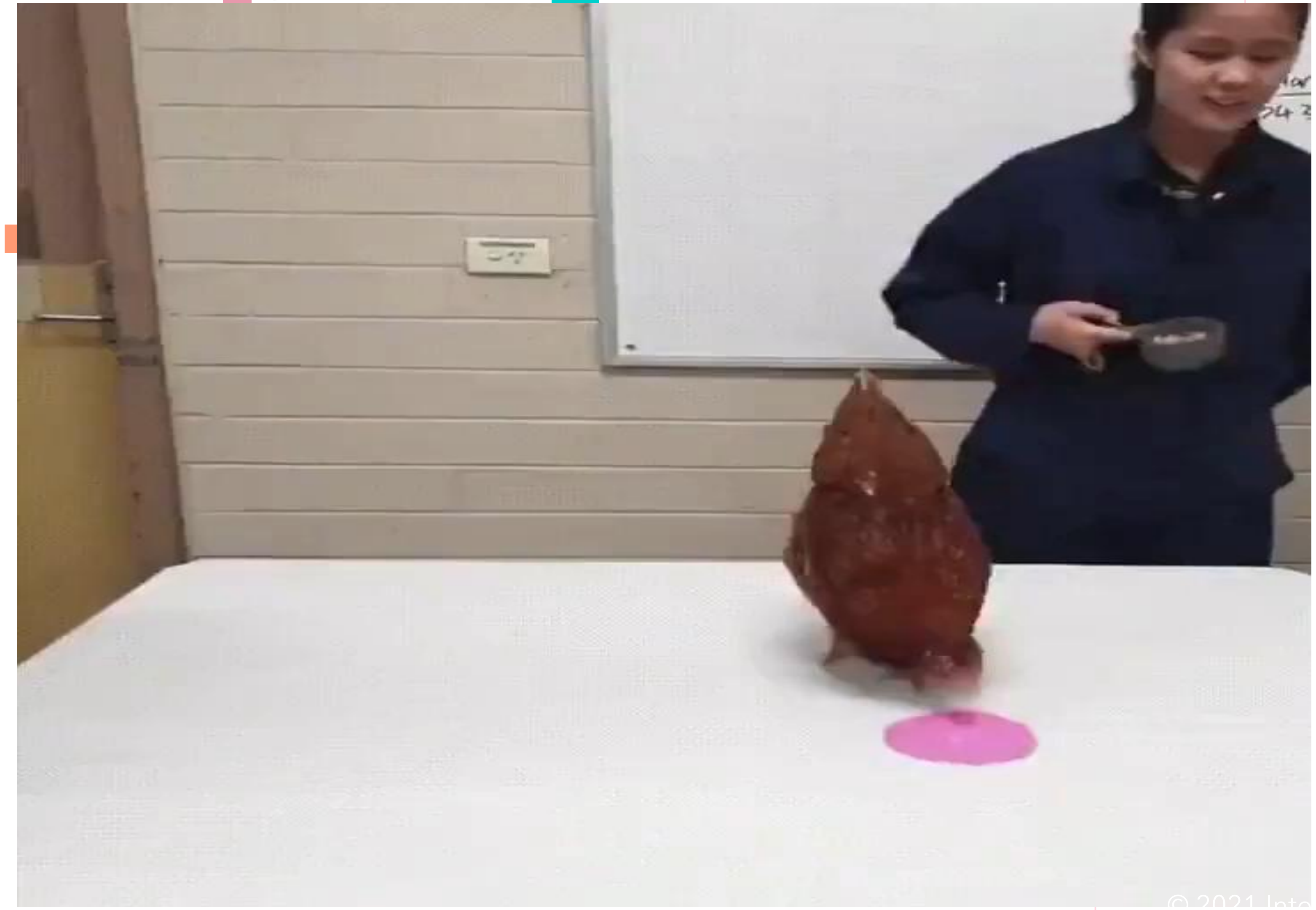
"The researchers at DeepMind suggest reinforcement learning as the main algorithm that can replicate reward maximization as seen in nature and can eventually lead to artificial general intelligence." https://venturebeat.com/ai/deepmind-says-reinforcement-learning-is-enough-to-reach-general-ai/#:~:text=Reinforcement%20learning%20is%20a%20special,and%20that%20of%20the%20environment.

AlphaGo is a computer program that plays the board game Go. It was developed by the London-based DeepMind Technologies. It does use reinforcement learning (RL) - in conjunction with supervised learning and Monte Carlo Tree Search (MCTS)

What is learning?

- Learning takes place as a result of interaction between an agent and the world, the idea behind learning is that
  - Perception received by an agent should be used not only for acting, but also for improving the agent's ability to behave optimally in the future to achieve the goal.
- Unsupervised, Supervised & Reinforcement Learning

Let's take a break & enjoy watching this short video

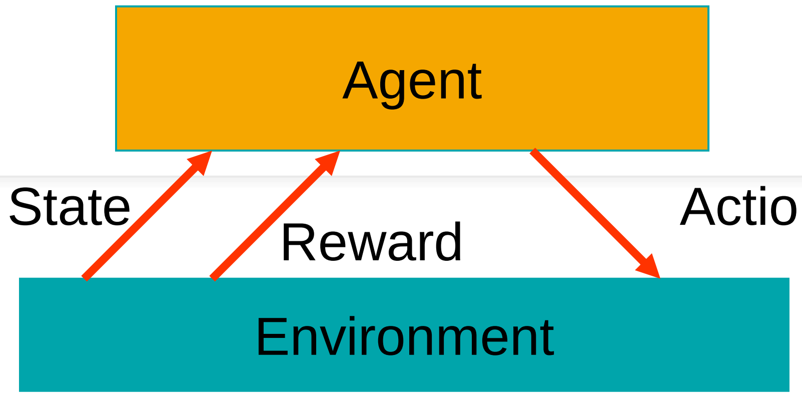


Reinforcement Learning

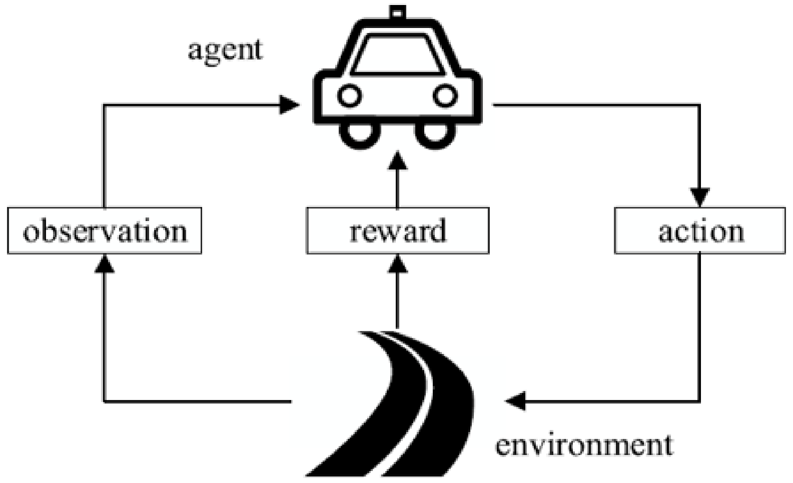
Reinforcement learning is a machine learning training method based on rewarding desired behaviours and/or punishing undesired ones.

In general, a reinforcement learning agent is able to perceive and interpret its environment, take actions and learn through trial and error.

RL PROCESS



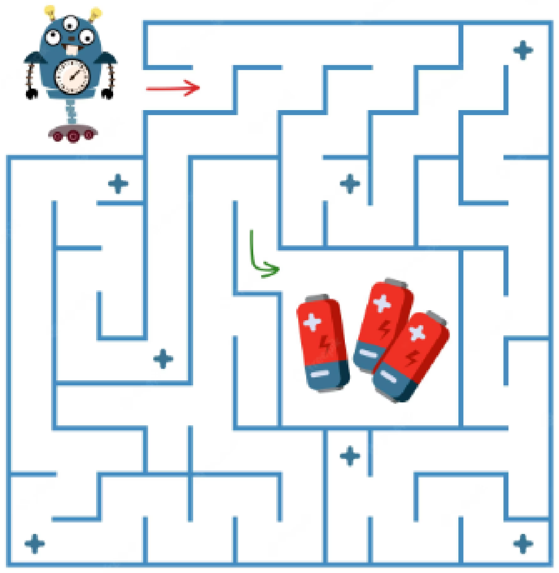
$$\begin{tikzcd} s_0 \arrow[r, "a_0", "r_0"'] & s_1 \arrow[r, "a1", "r1"'] & s2 \arrow[r, "a2", "r2"'] & s3 \end{tikzcd}$$



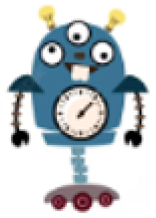
- Agent-environment interaction: the agent selects an action based on its policy at each time step after observing the surroundings. Following the completion of the activity, the environment transitions to a new state and sends the agent a reward signal

Agent → Action → Environment → State → Reward → Agent.

The Robot is looking for the battery to recharge!

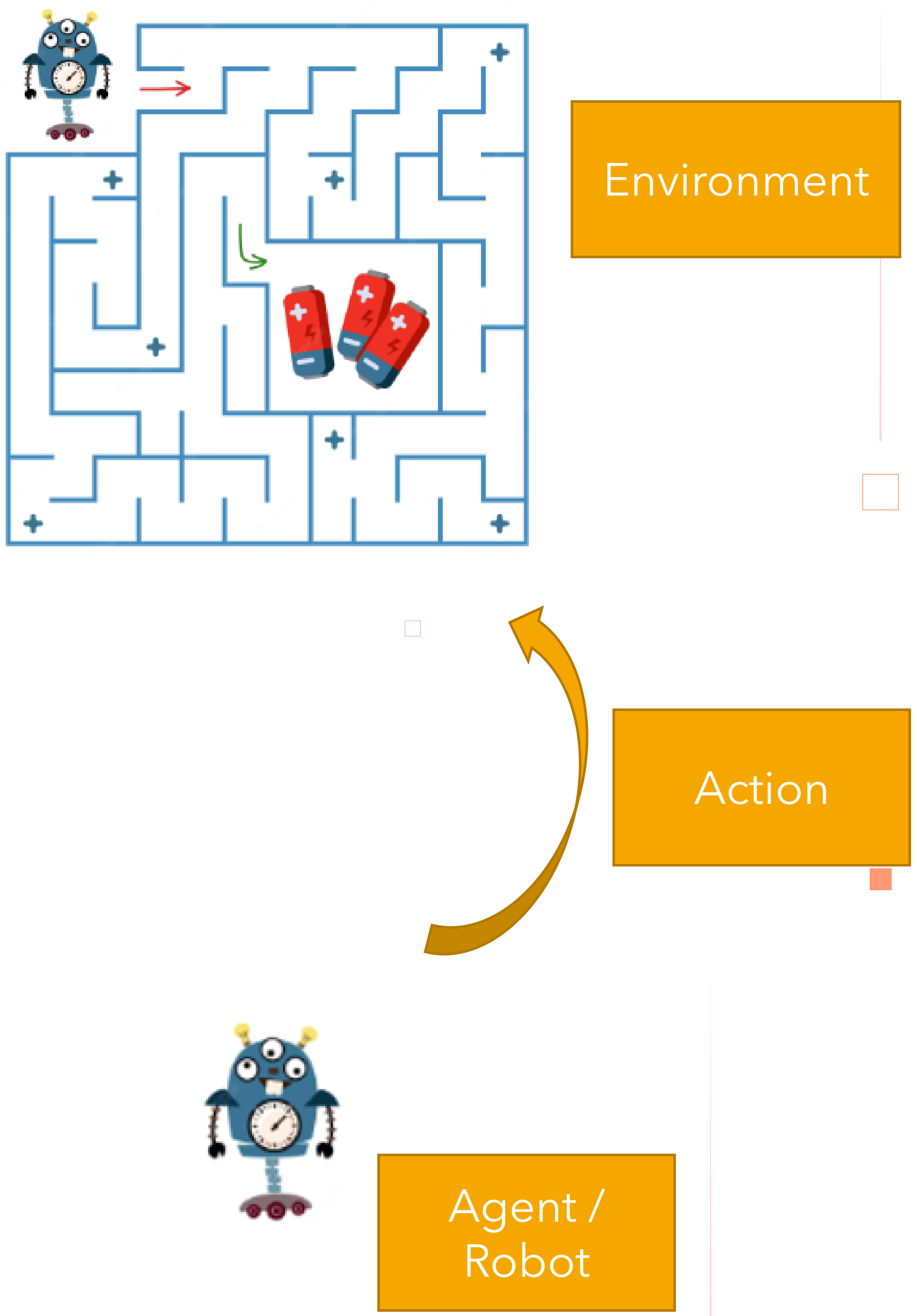


Environment

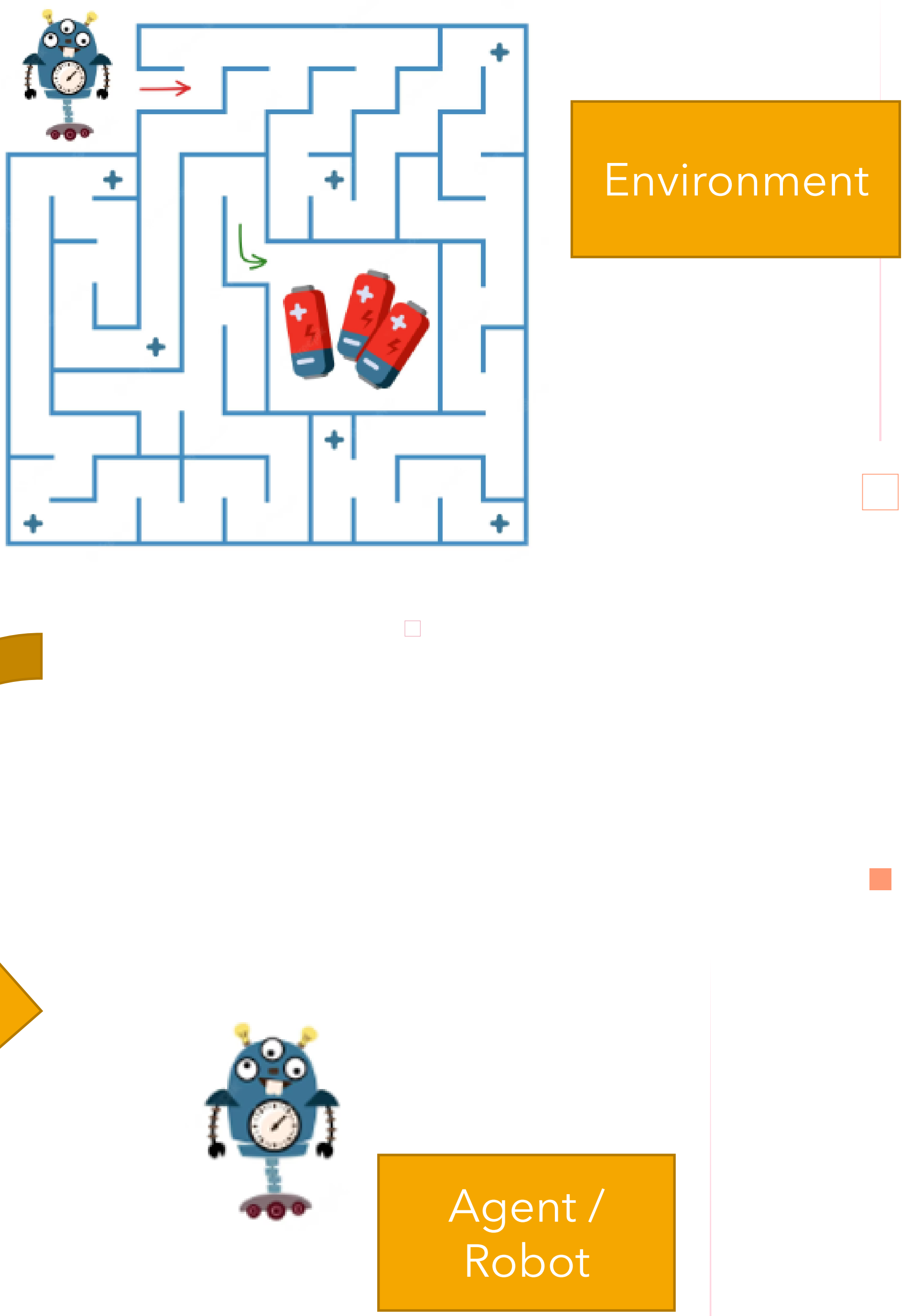


Agent / Robot

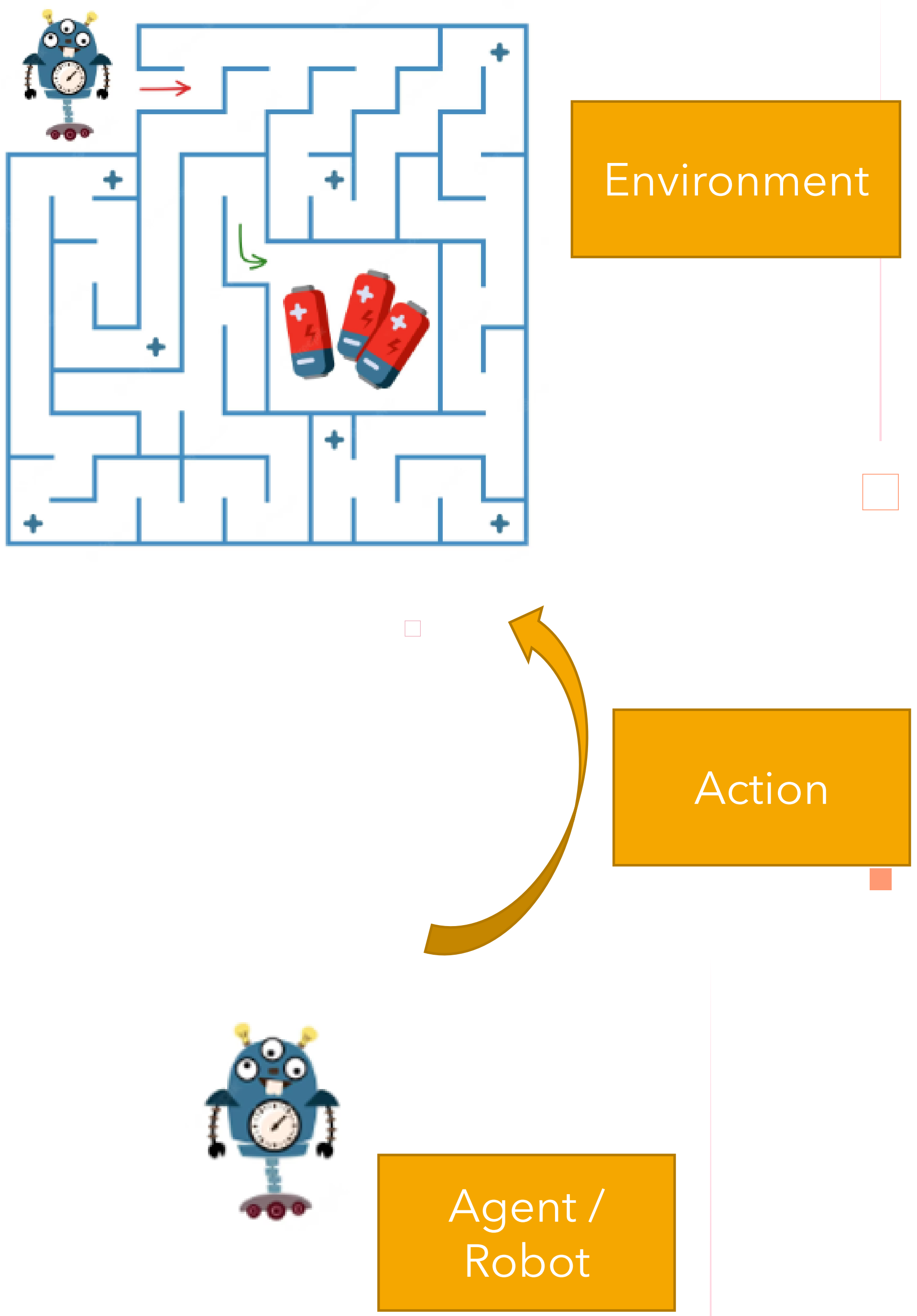
The Robot is looking for the battery to recharge!



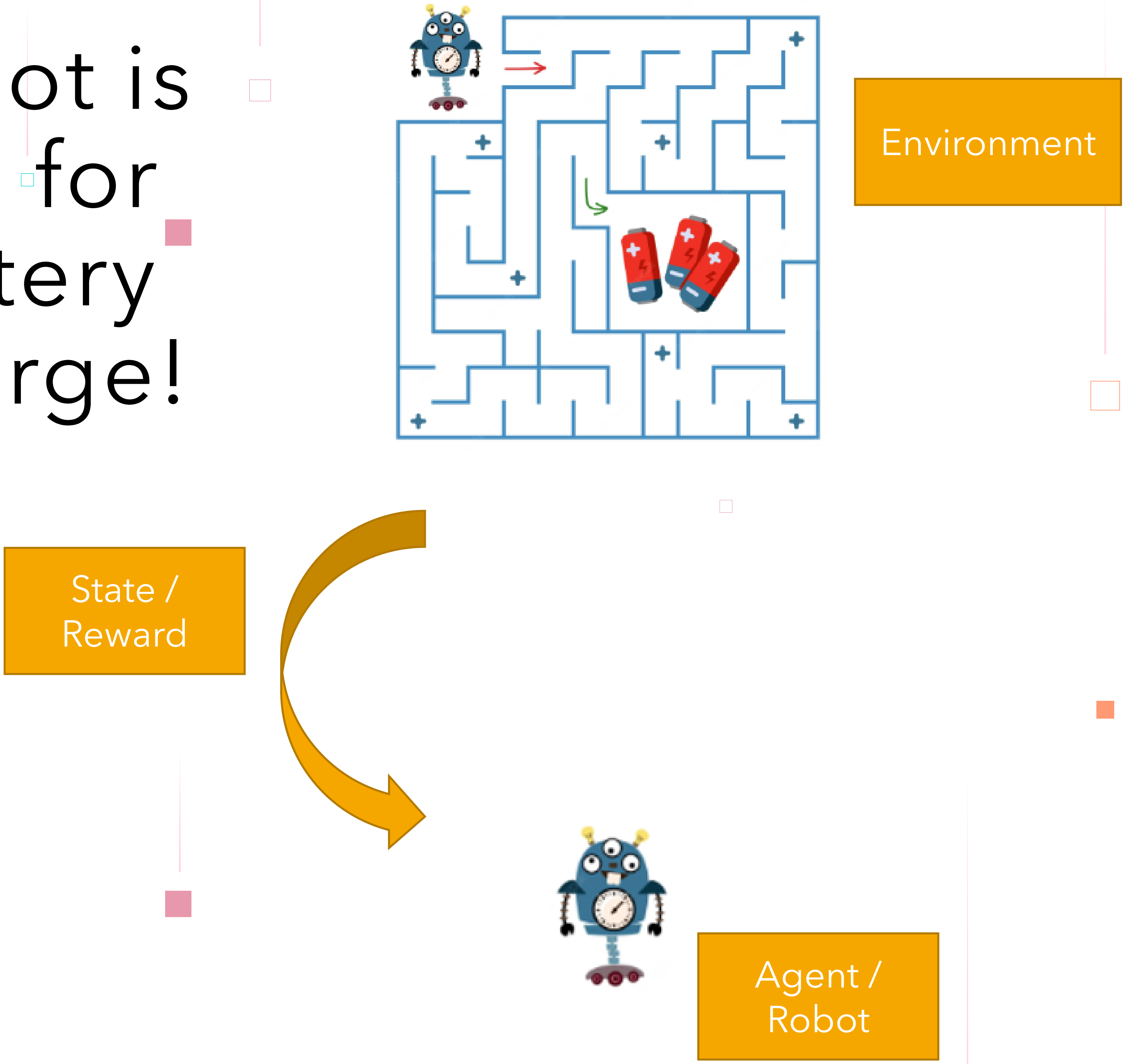
The Robot is looking for the battery to recharge!



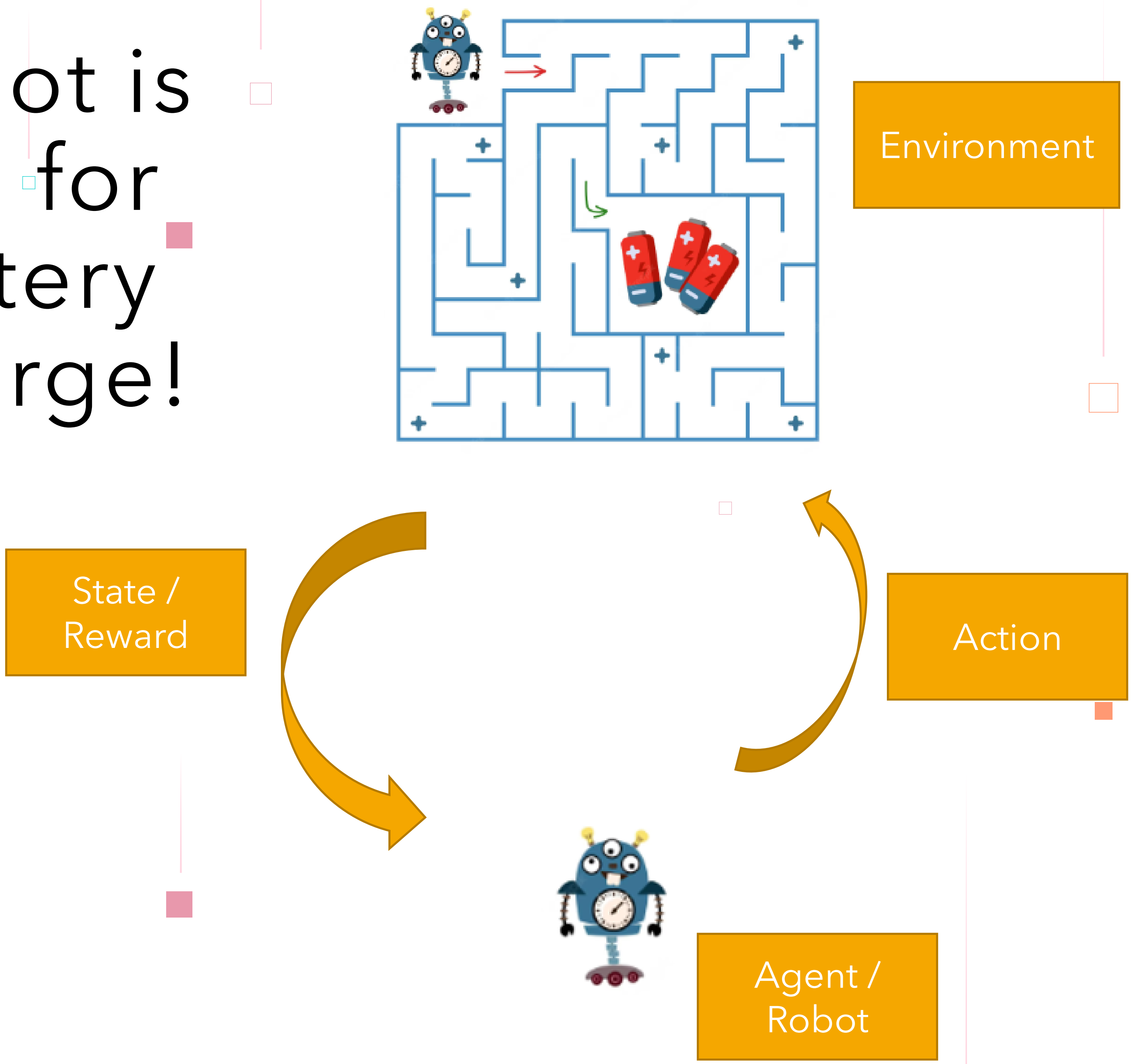
The Robot is looking for the battery to recharge!



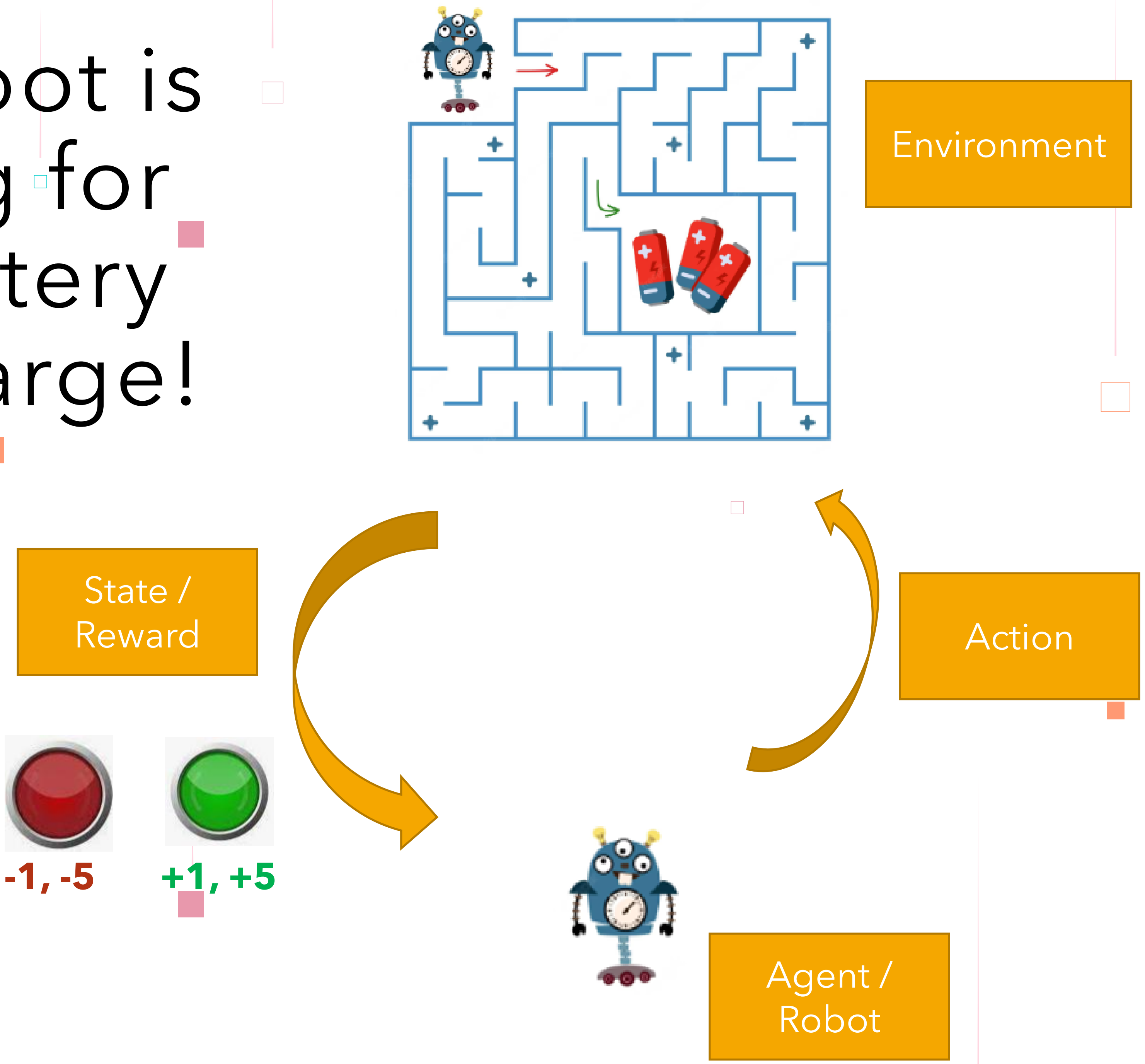
The Robot is looking for the battery to recharge!



The Robot is looking for the battery to recharge!



The Robot is looking for the battery to recharge!

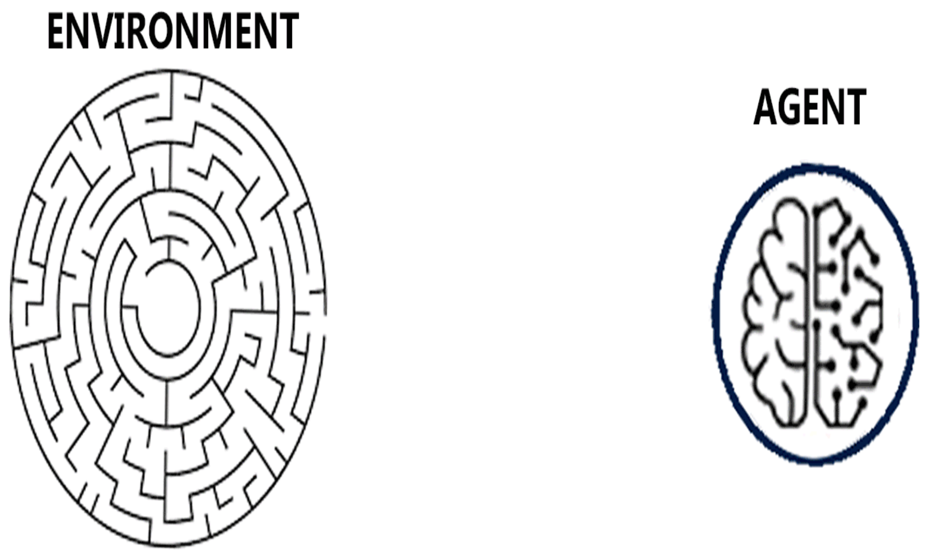


Reinforcement Learning

Learning by trial-and-error through reward or punishment.

The program learns by playing the game millions of times. We reward the program when it makes a good move. This strengthens the connections to make moves like it did. When it loses, we give no reward (or negative reward).

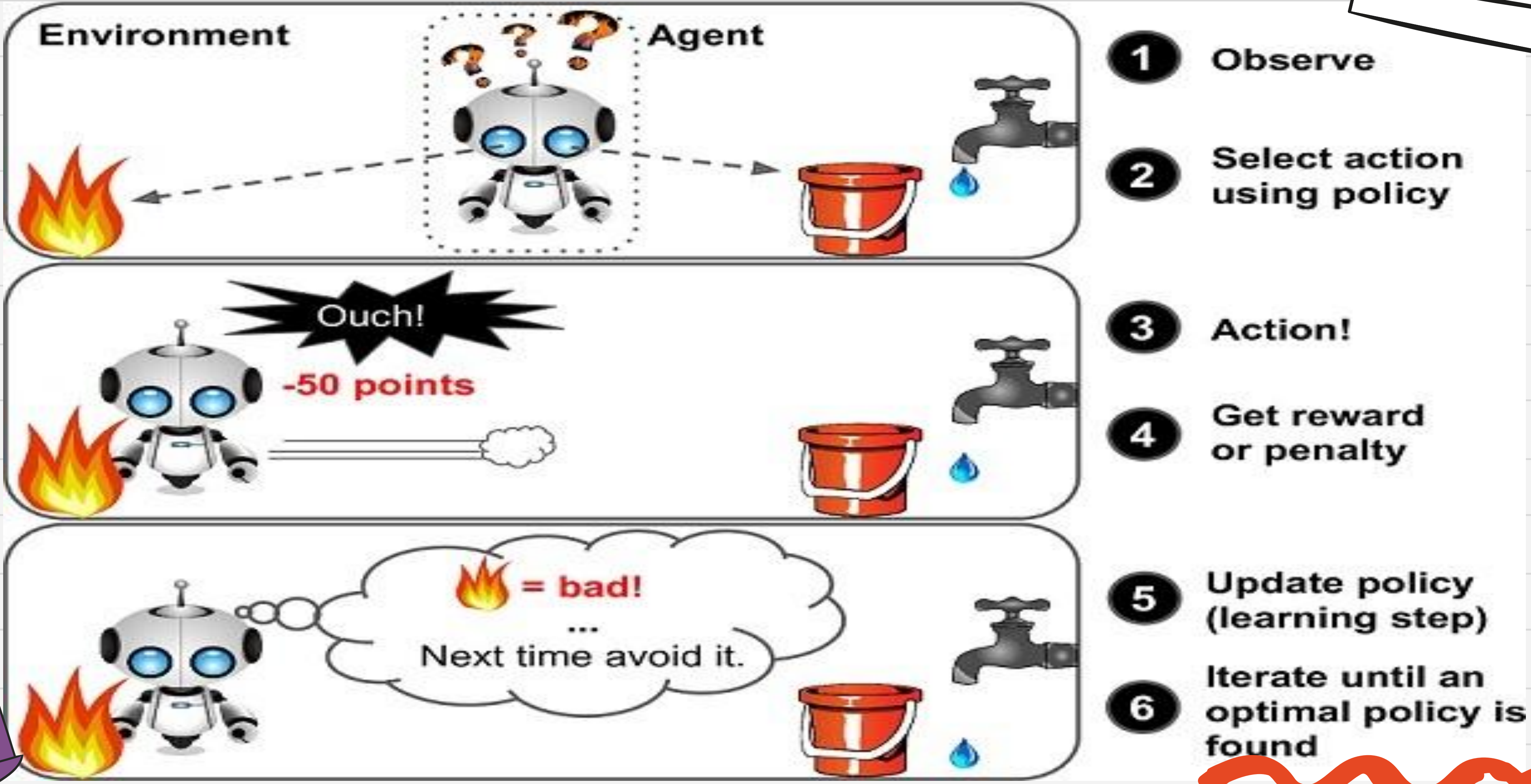
Over time it learns to maximize reward without the human explicitly telling the rules. It can lead to better than human performance when it finds plays that no one ever thought of doing before…



More insights!-meet-Xiaomi CyberOne

Reinforcement Learning

Rewards from sequence of actions



1 Observe

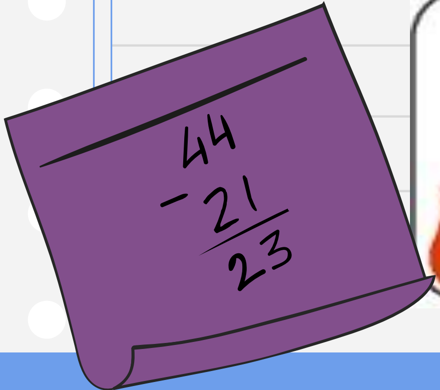
2 Select action using policy

3 Action!

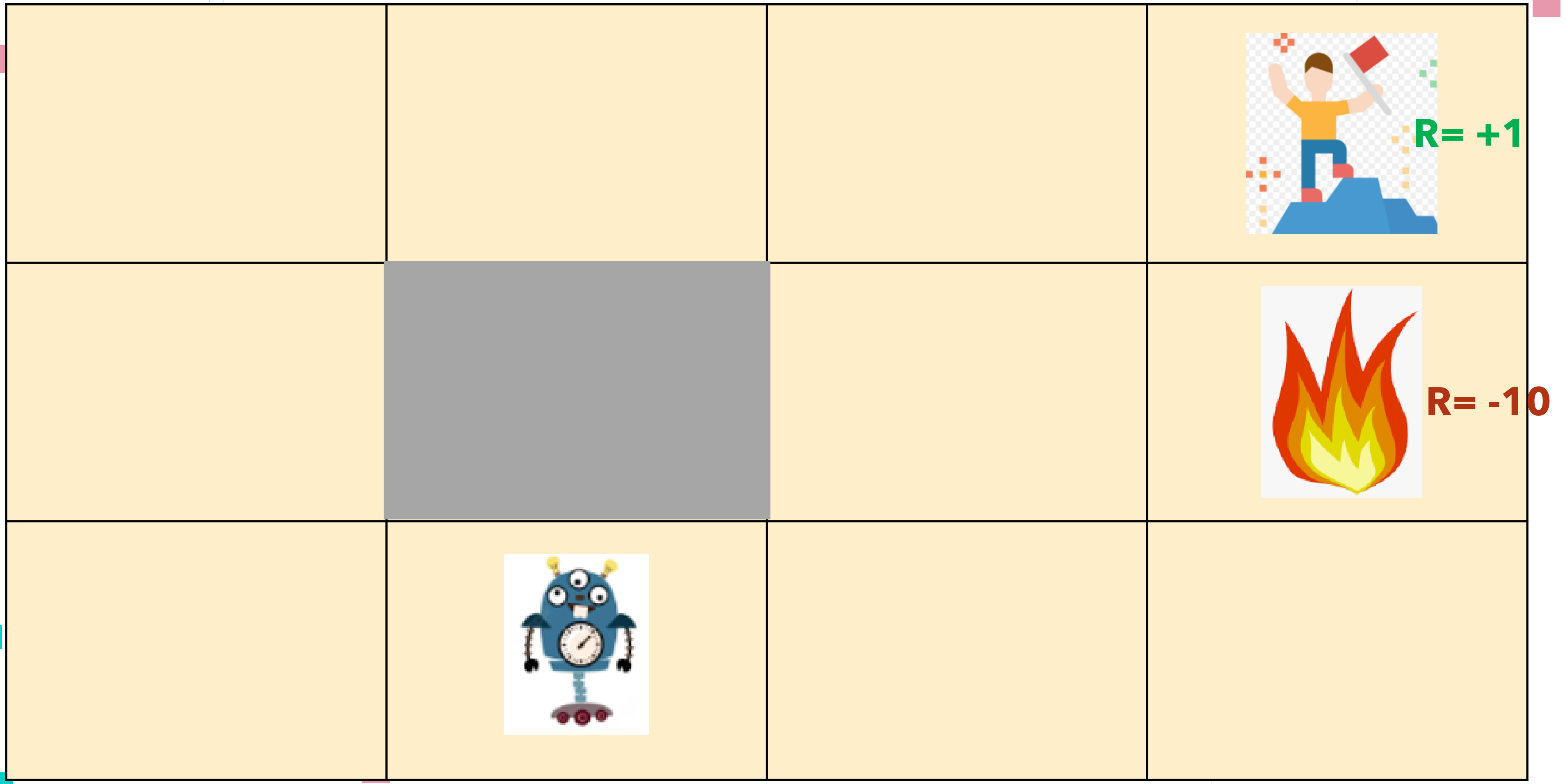
4 Get reward or penalty

5 Update policy (learning step)

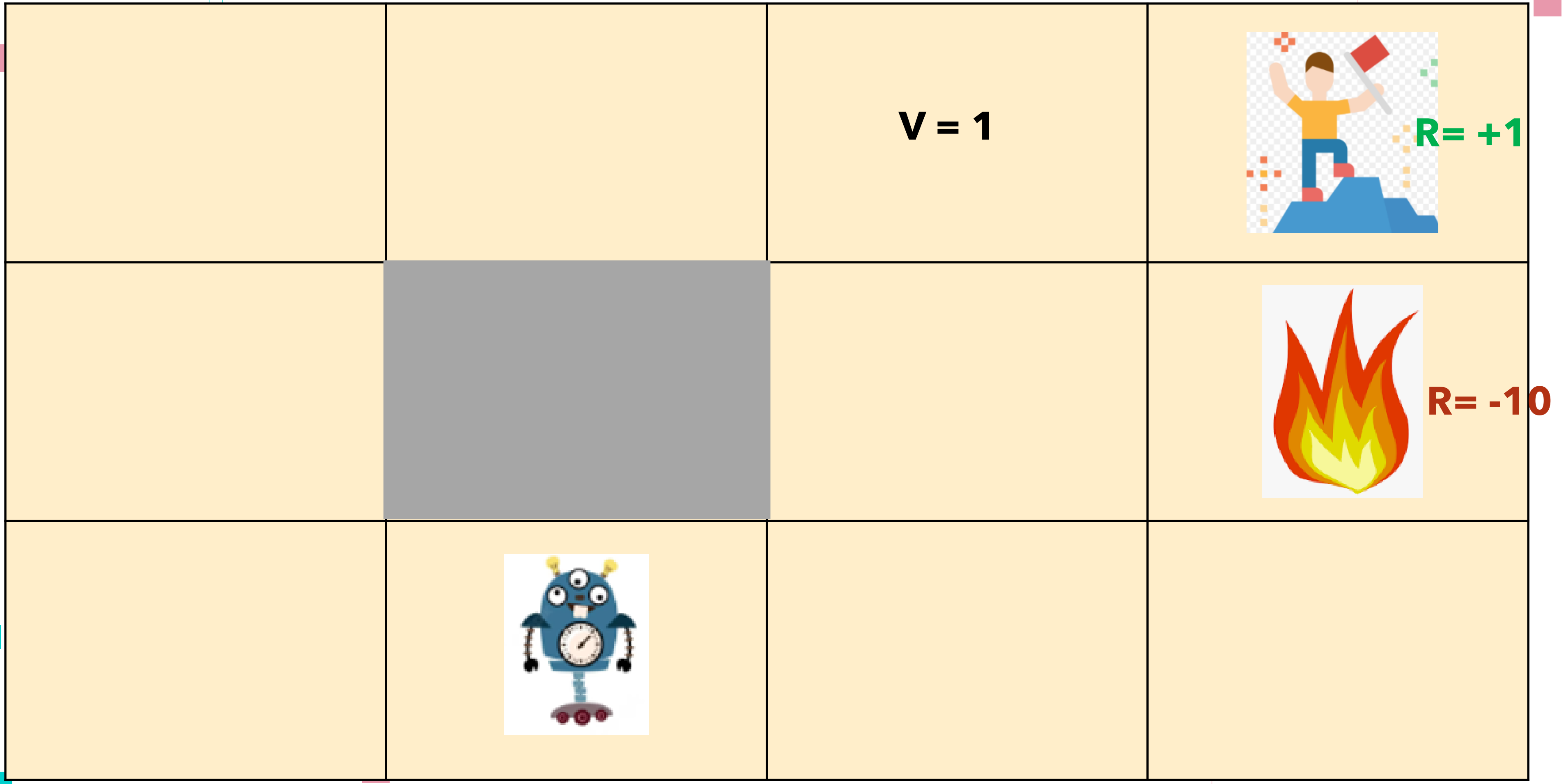
6 Iterate until an optimal policy is found



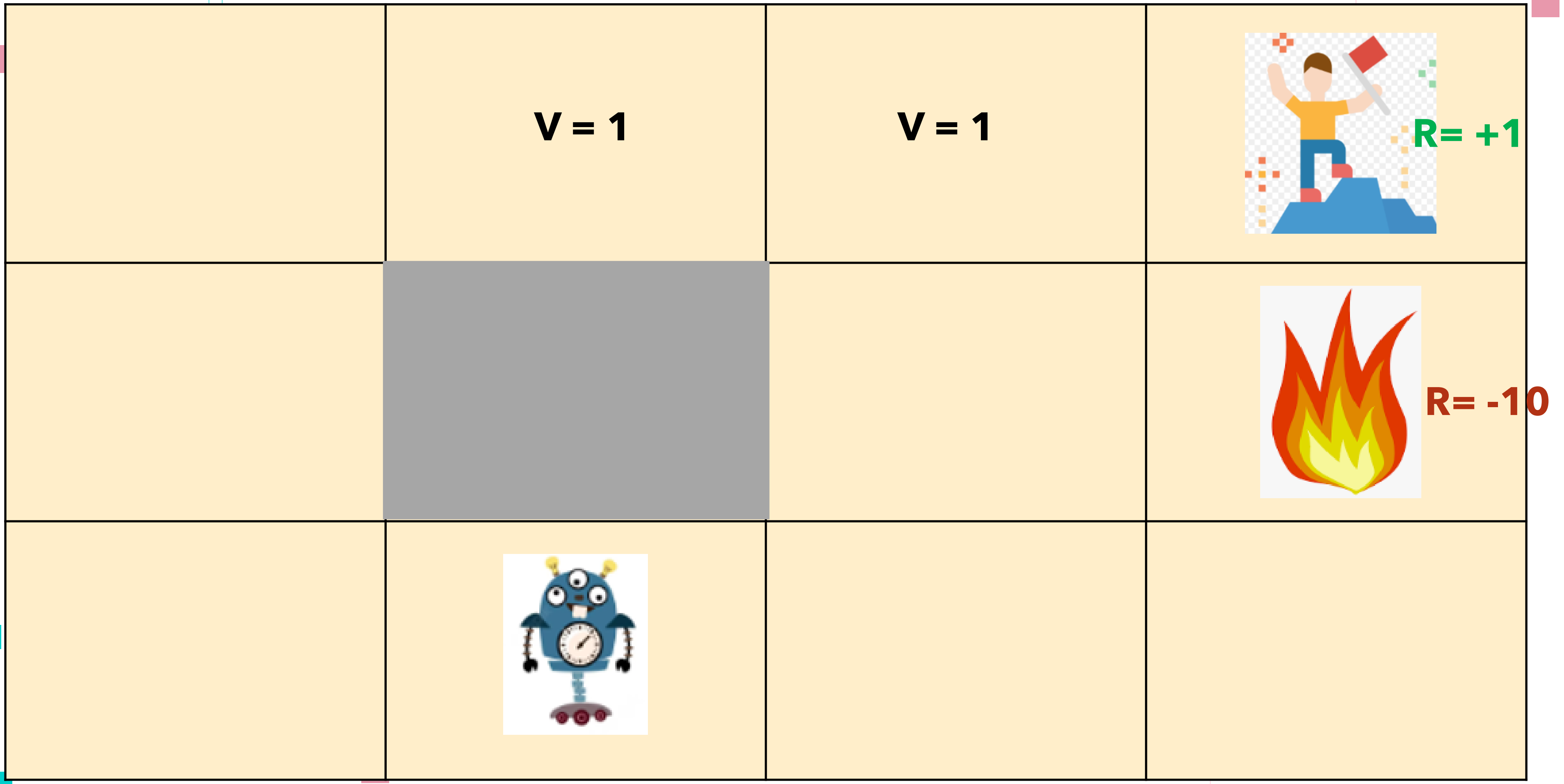
Basic Environment



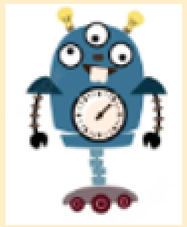
Basic Environment



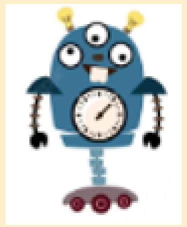
Basic Environment



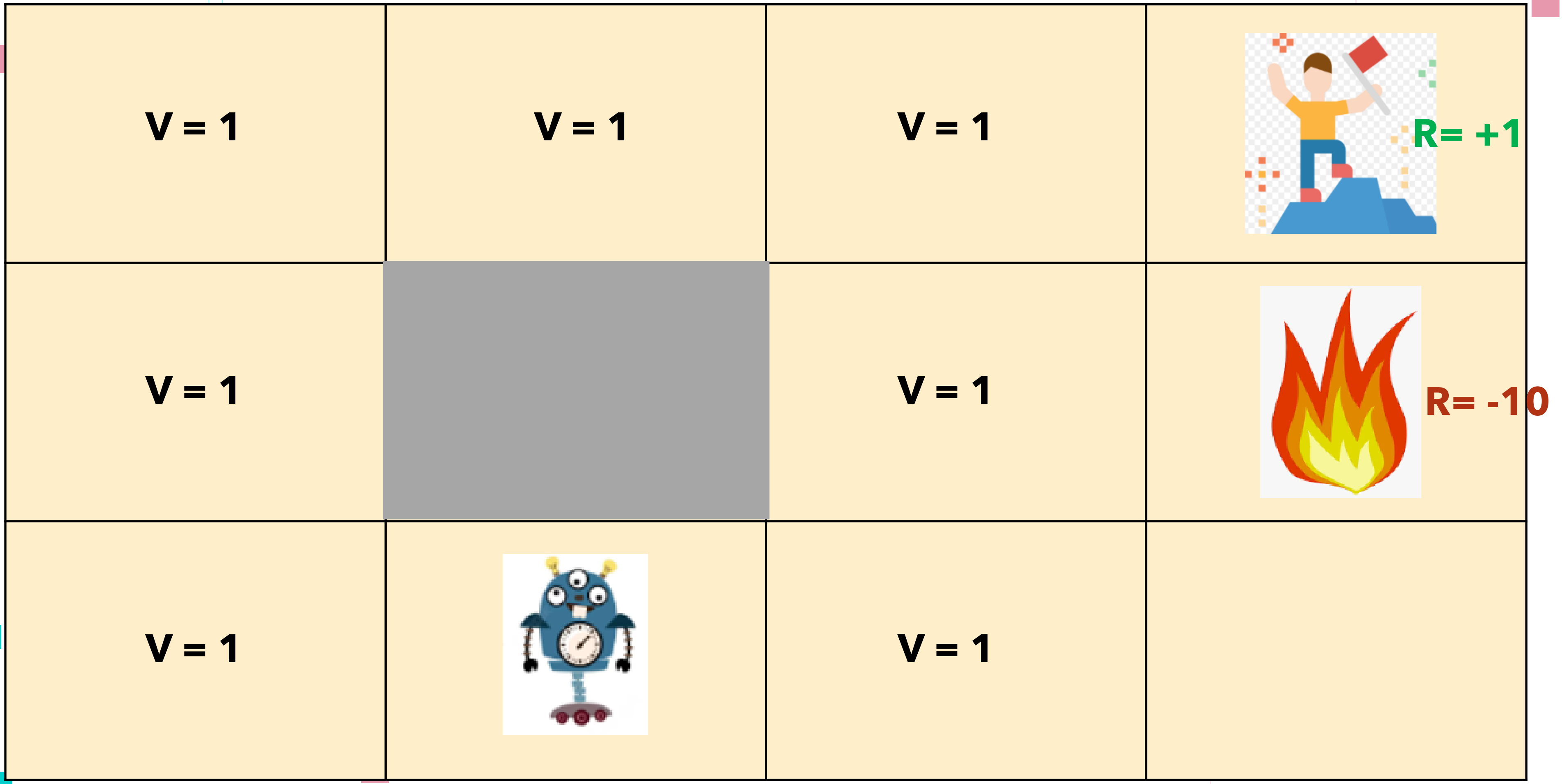
Basic Environment

V = 1	**V = 1**	**V = 1**	**R= +1**
V = 1	▨	**???**	**R= -10**
V = 1			

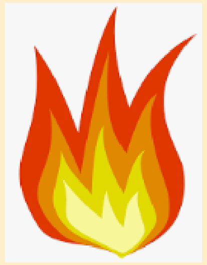
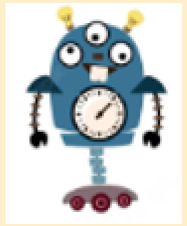
Basic Environment

V = 1	V = 1	V = 1	R= +1
V = 1	▨	V = 1	R= -10
V = 1			

Basic Environment

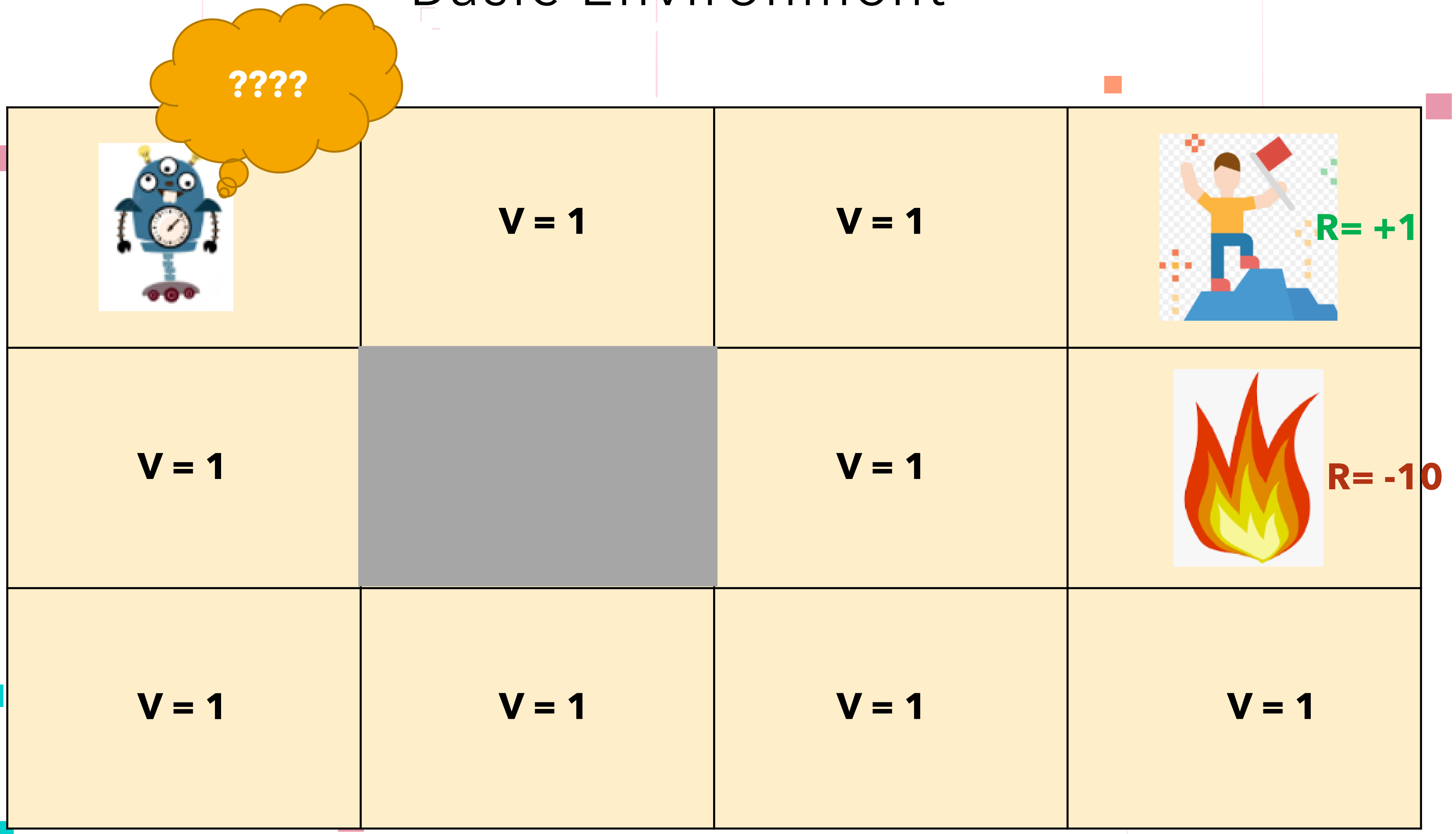


Basic Environment

V = 1	V = 1	V = 1	R= +1
V = 1	▨	V = 1	 R= -10
V = 1		V = 1	V = 1

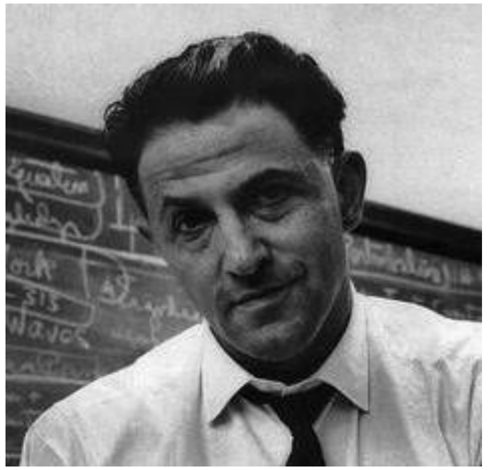
Is it Complete?

Basic Environment



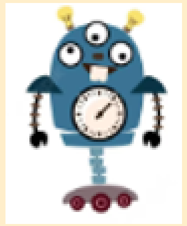
The Bellman Equation, 1953

- $V(s) = \max_a(R(s,a) + \gamma V(s'))$
- γ: discount factor, [0,1]
- $V(s')$: $value\ for\ next\ state$
- $R(s,a)$: Reward to take that action


Sir Richard Ernest Bellman

Basic Environment

$$V(s) = \max_a (R(s,a) + \gamma V(s'))$$
$$\gamma = 0.9$$

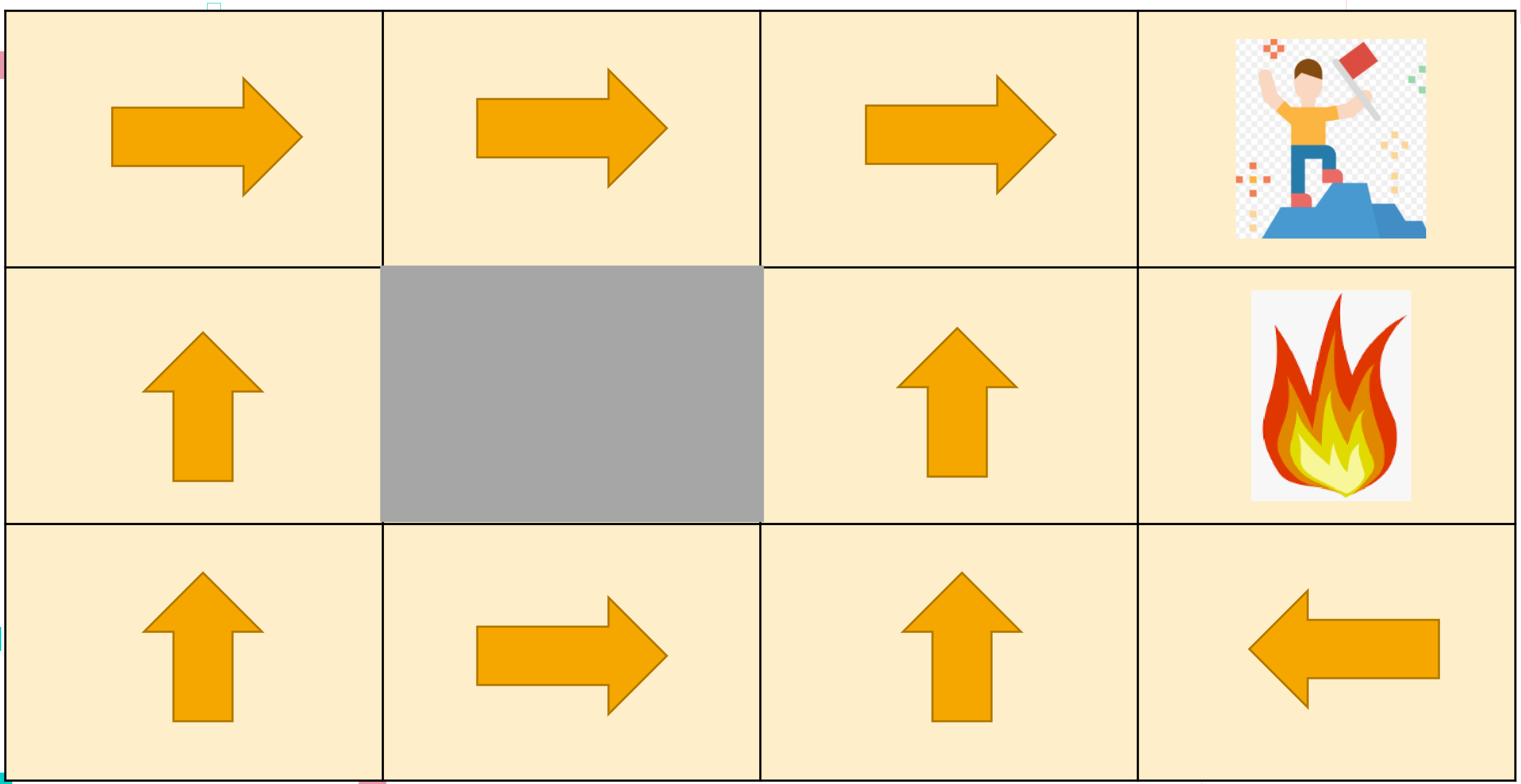
V = 0.81	**V = 0.9**	**V = 1**	**R= +1**
V = 0.73	▨		**R= -10**
V = 0.66			

Basic Environment

$$V(s) = \max_a (R(s,a) + \gamma V(s'))$$
$$\gamma = 0.9$$

V = 0.81	V = 0.9	V = 1	R= +1
V = 0.73	▨	V = 0.9	R= -10
V = 0.66	V = 0.73	V = 0.81	V = 0.73

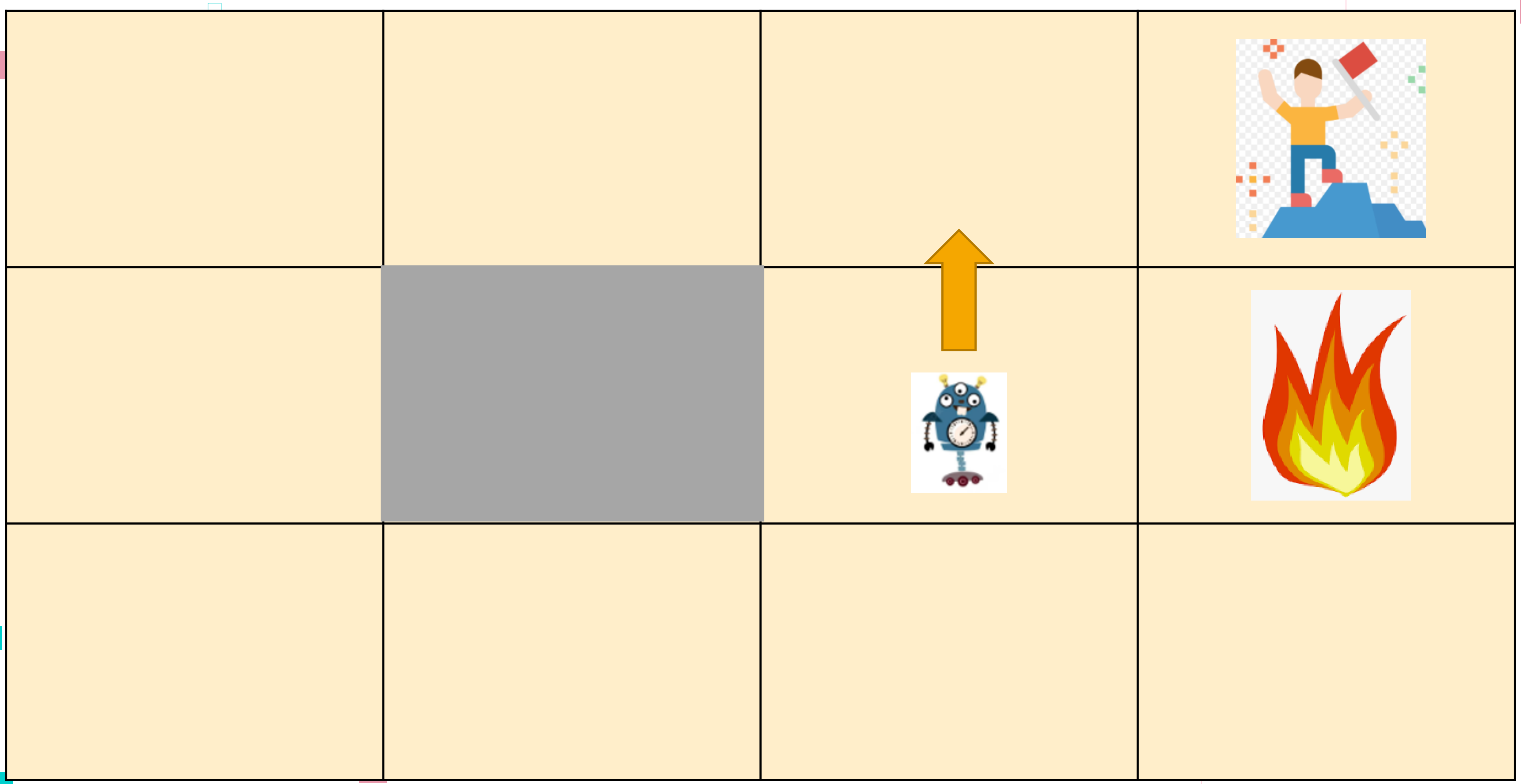
The Plan



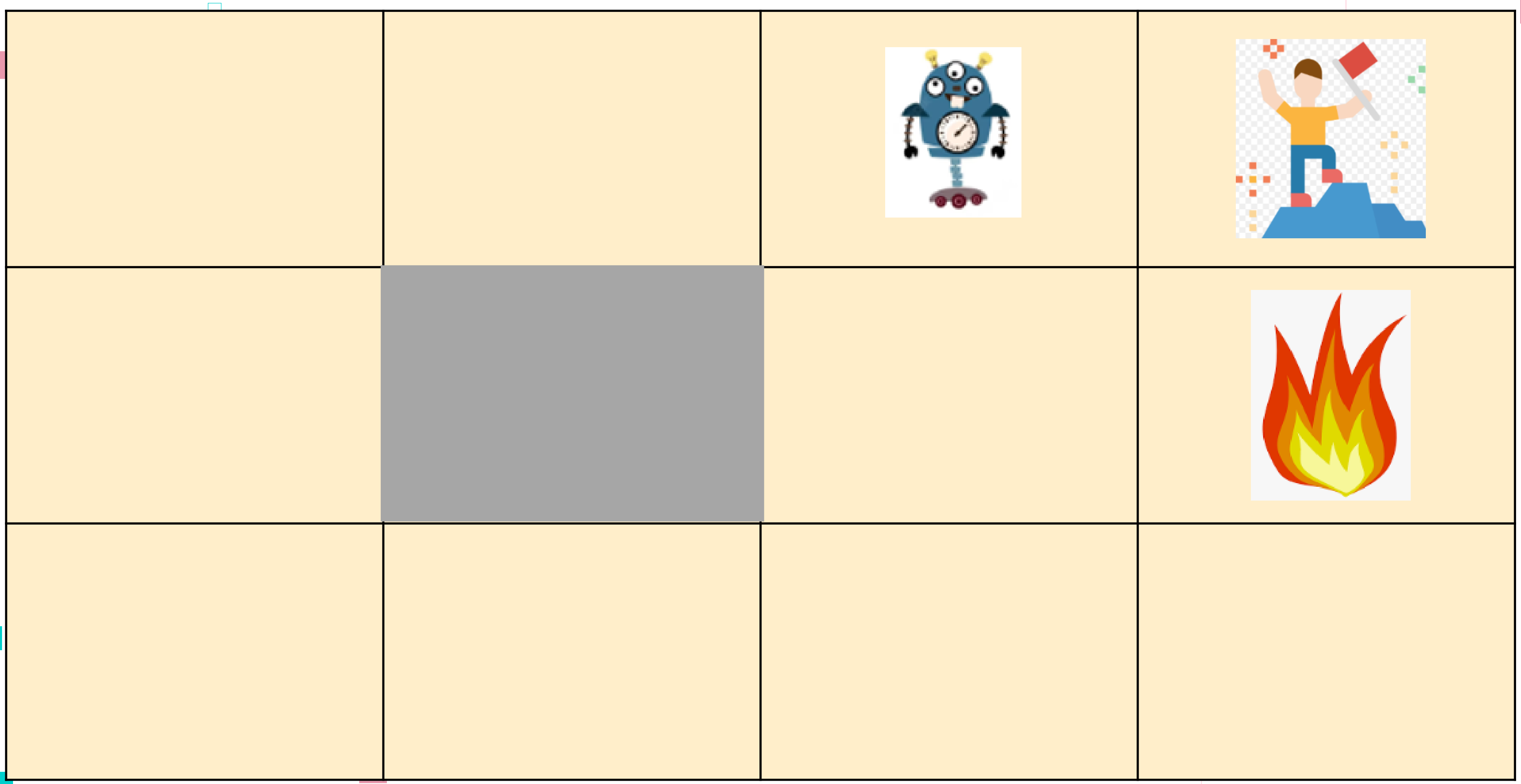
Plan vs Policy: The Plan

- A plan is a sequence of actions designed to achieve a specific goal or reach a particular state from a starting state.
- Characteristics:
  - Sequential: A plan outlines a predefined sequence of actions to be executed.
  - Goal-Oriented: Plans are devised with the objective of reaching a specific goal or target state.
  - Time-Bound: Plans are often developed for a specific timeframe or set of steps.
  - Complexity: Developing a plan may involve predicting the outcomes of actions over time, making it computationally intensive.
- Example:In a navigation task, a plan might involve a sequence of actions such as "move forward," "turn left," "move forward," until reaching the destination.

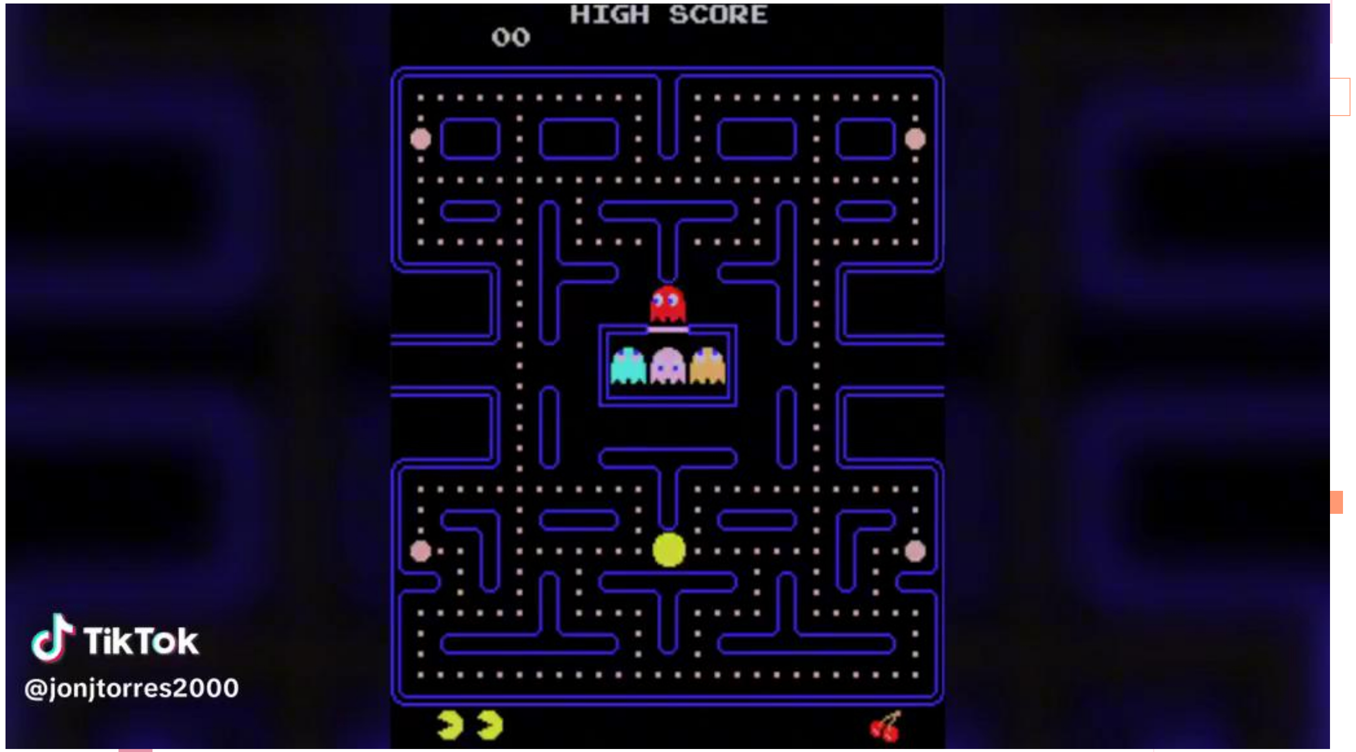
Deterministic Scenario



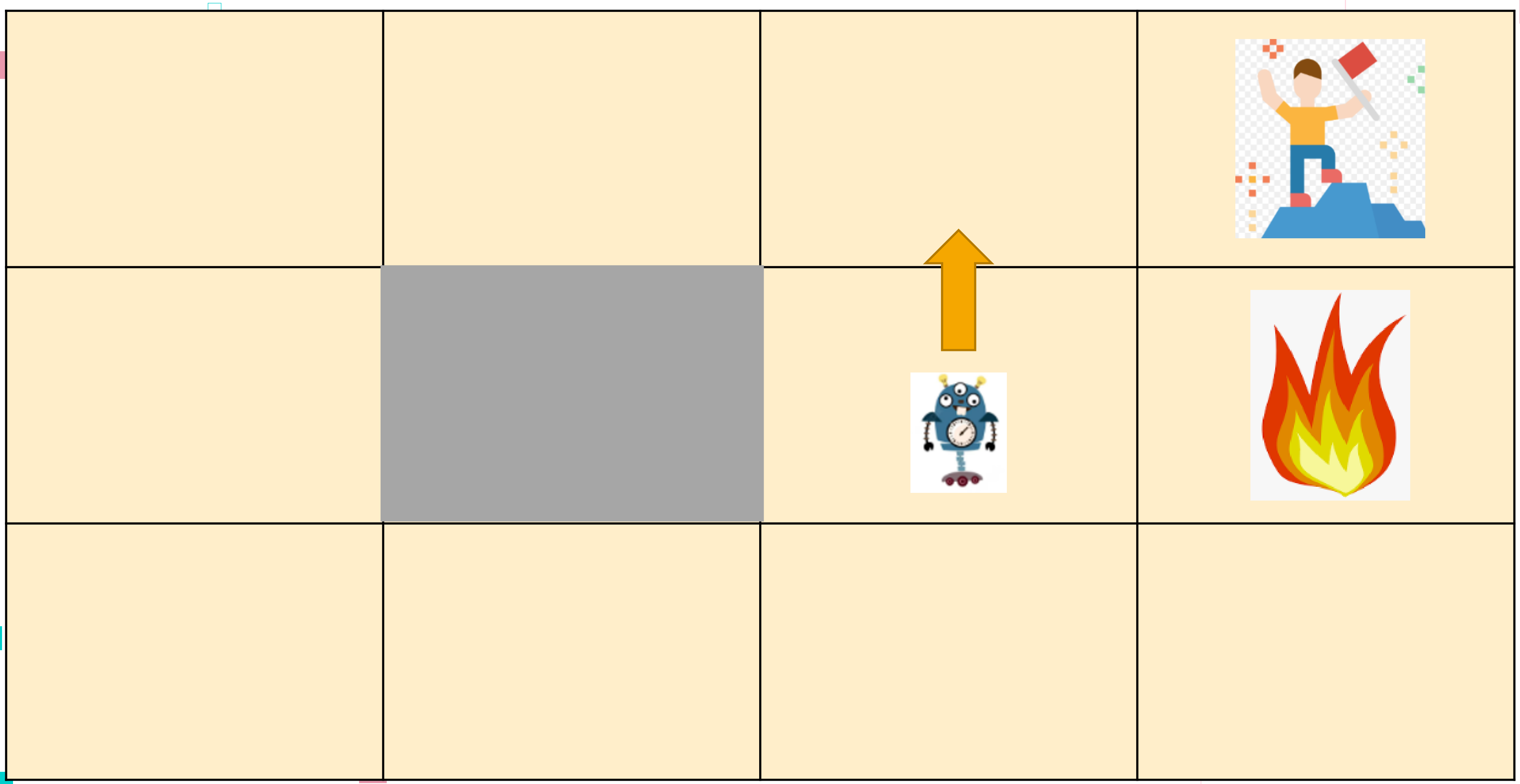
Deterministic Scenario



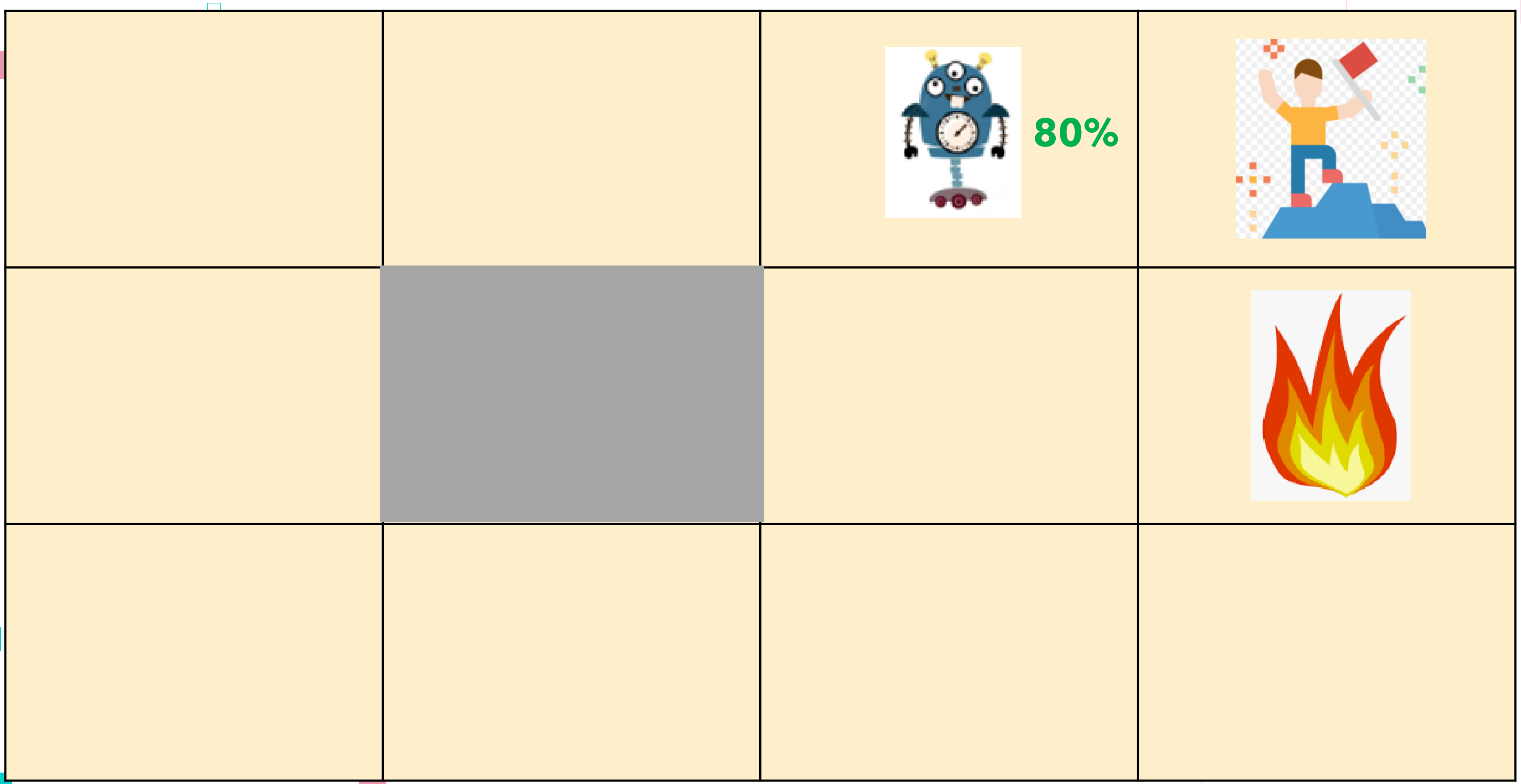
Watch this:



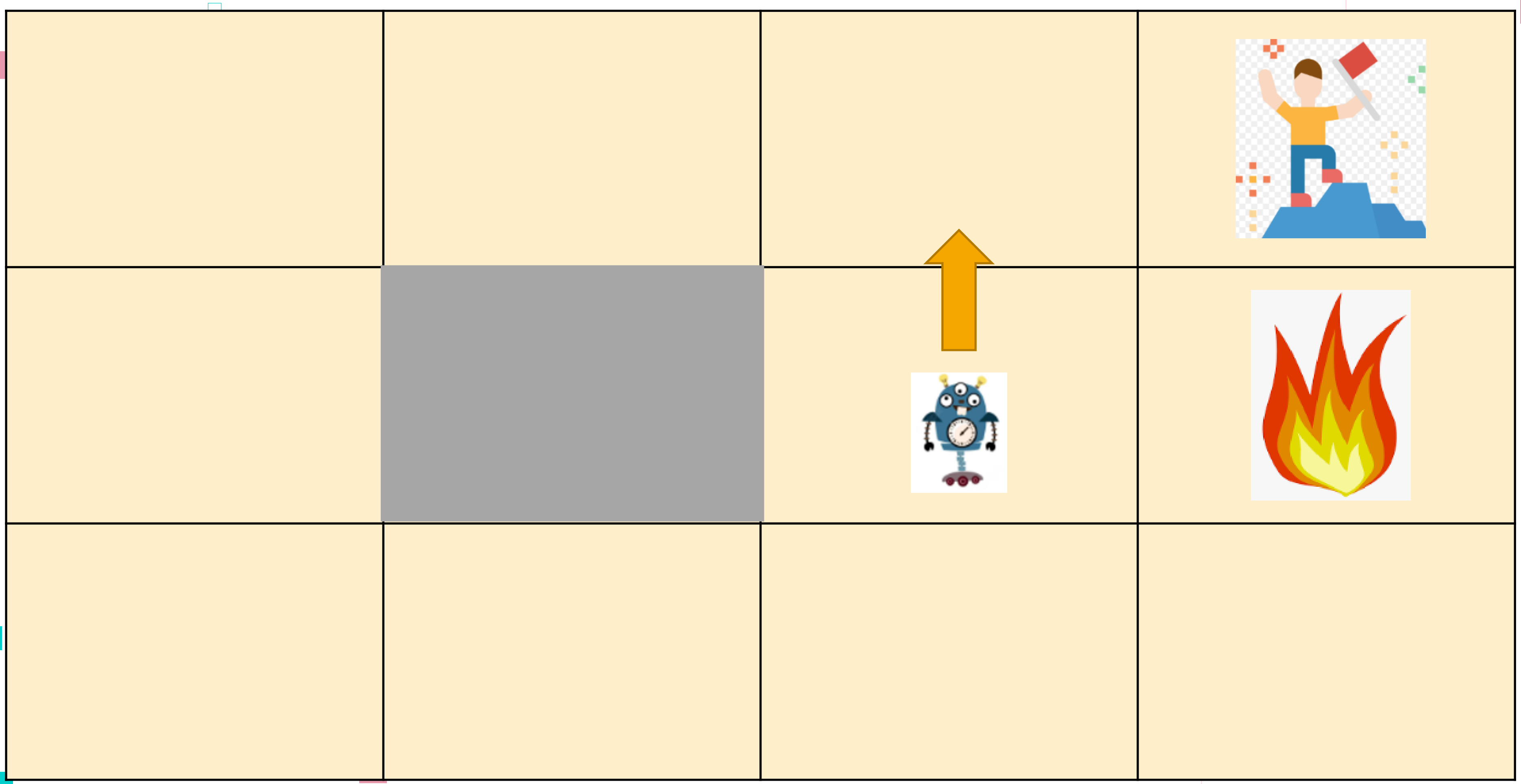
Actual Scenario - Randomness



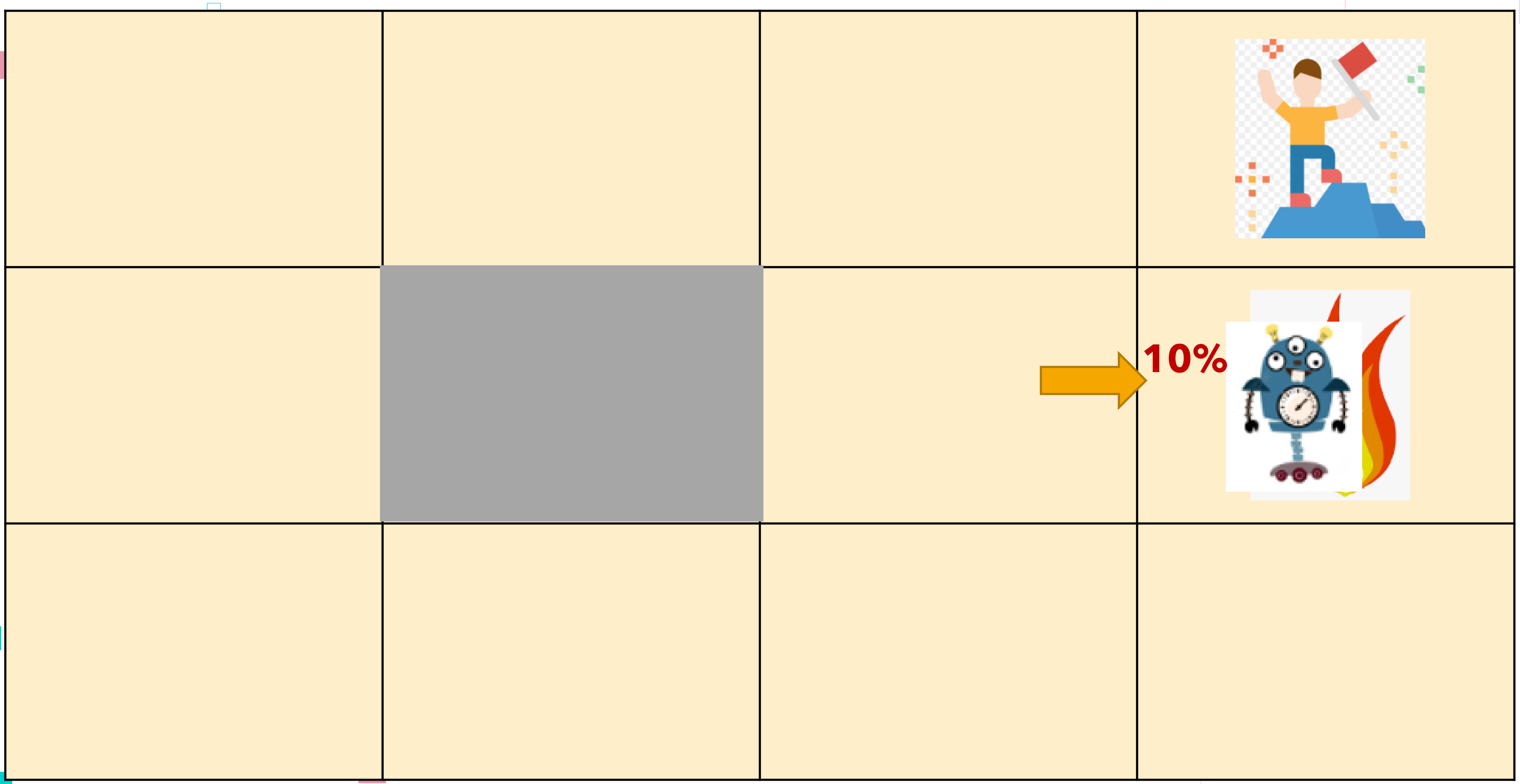
Actual Scenario - Randomness



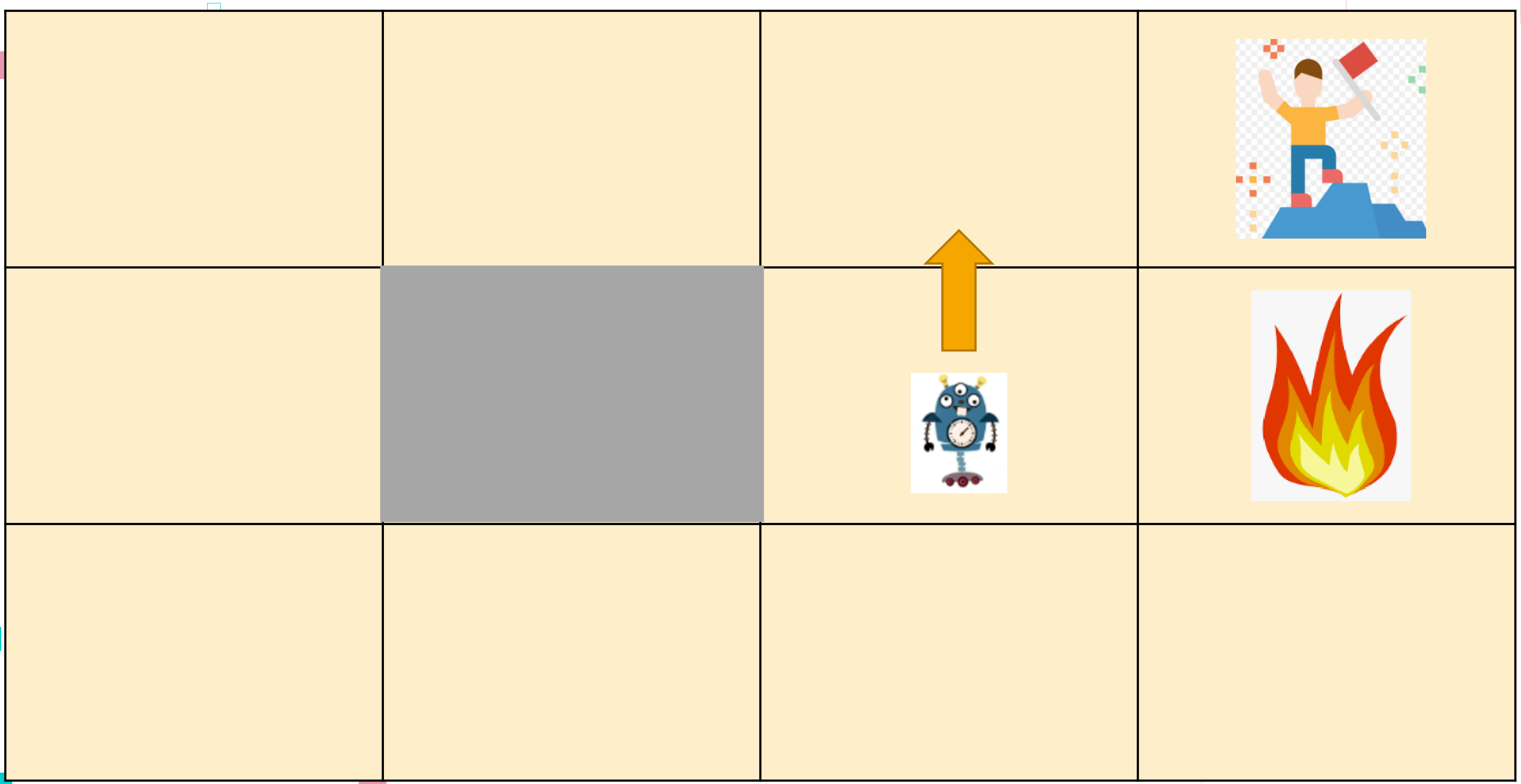
Actual Scenario - Randomness



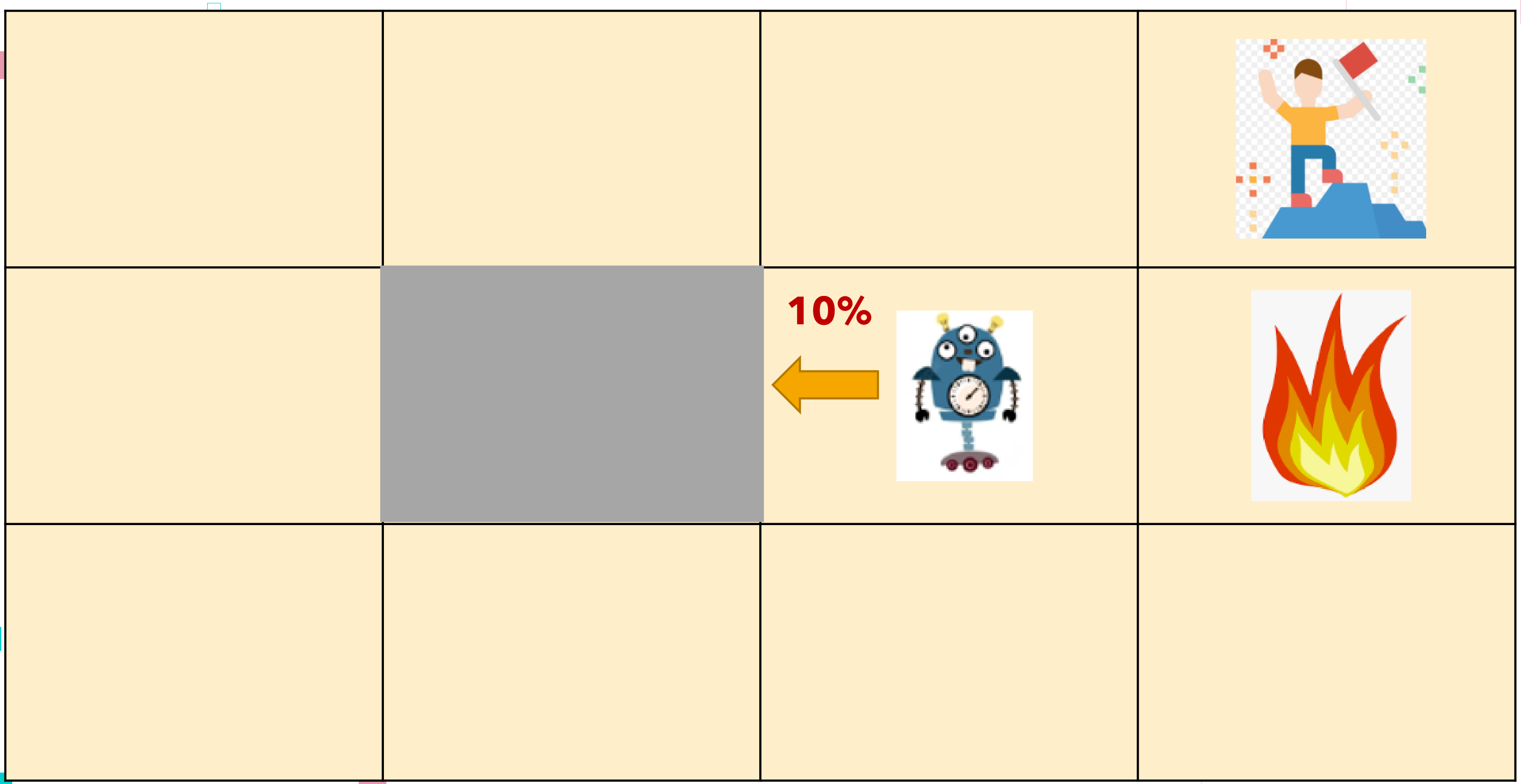
Actual Scenario - Randomness



Actual Scenario - Randomness

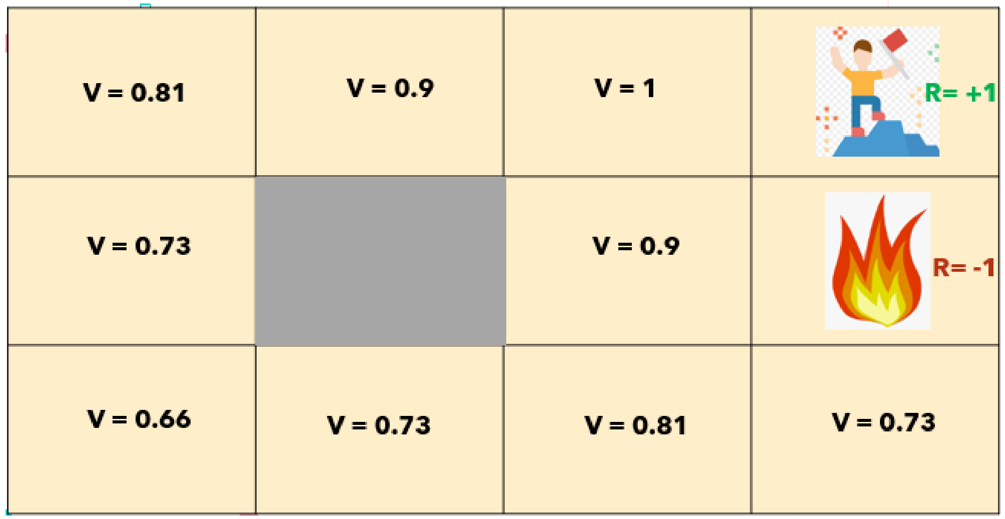


Actual Scenario - Randomness



Markov Decision Process (MDP)

- $V(s) = \max_a (\gamma \sum_{s'} P(s'|s,a)\,[R(s,a,s') + V(s')]$
- γ: discount factor, [0,1]
- $V(s')$: *value for next state*
- $R(s,a,s')$: The immediate reward received after transitioning from state s to state s' using action a.
- $P(s'|s,a)$: The transition probability of reaching state s' from state s when action a is taken [0.8(1+1)+0.1(-10+0.72)+0.1(0.9+0)]

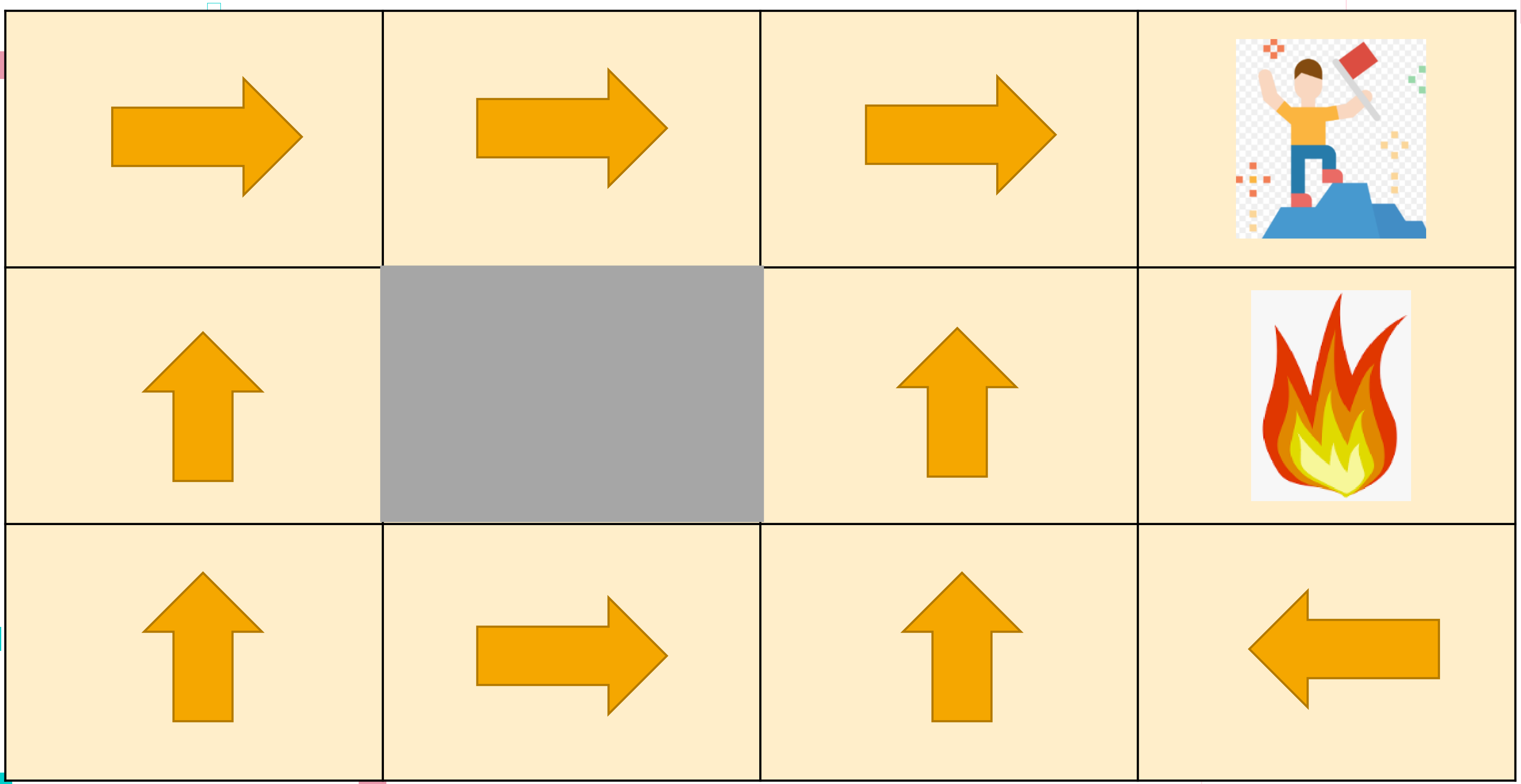


Policy vs Plan

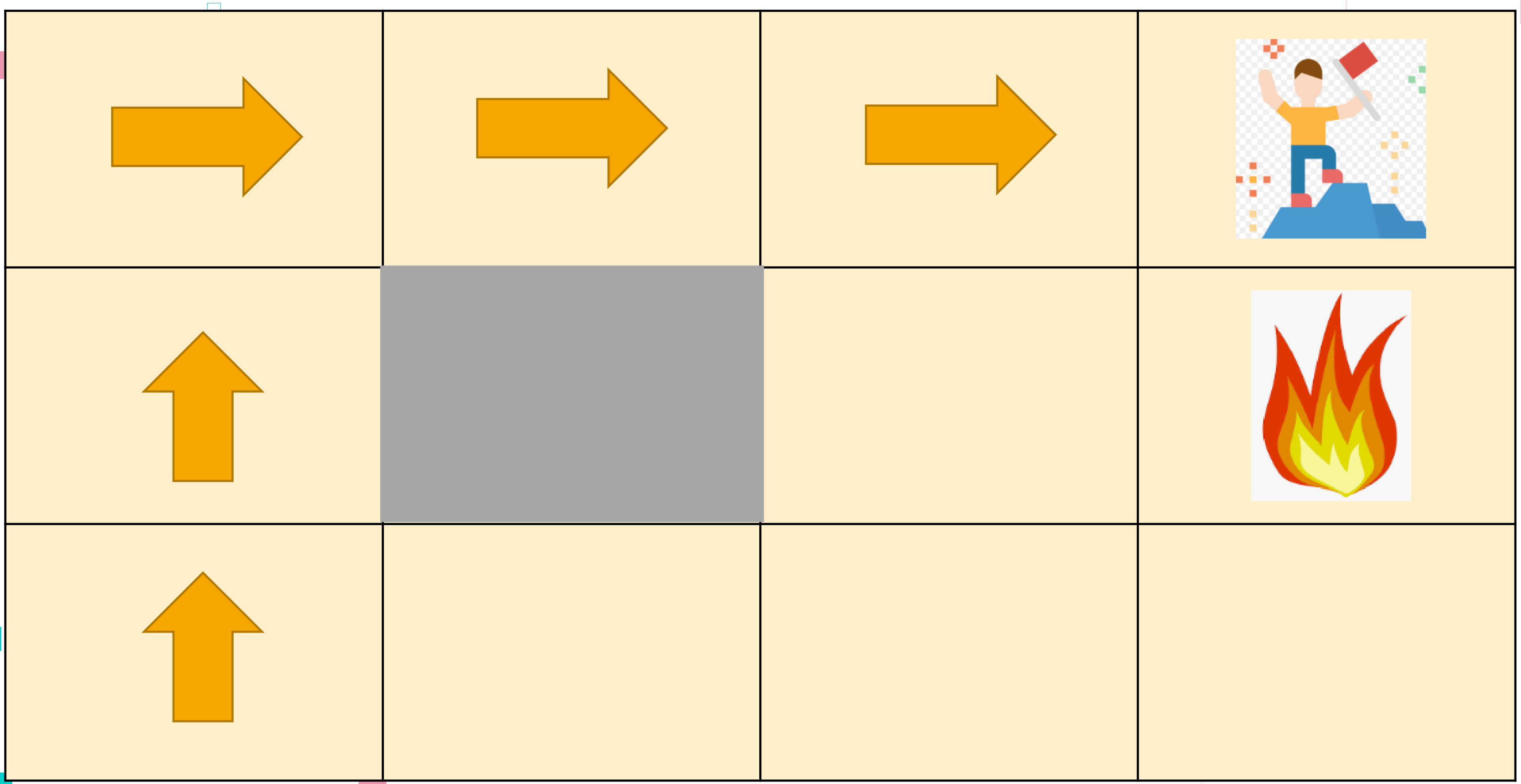
Let’s Guide the Robot

V = 0.81	V = 0.9	V = 1	R= +1
V = 0.73	▨	V = 0.9	R= -1
V = 0.66	V = 0.73	V = 0.81	V = 0.73

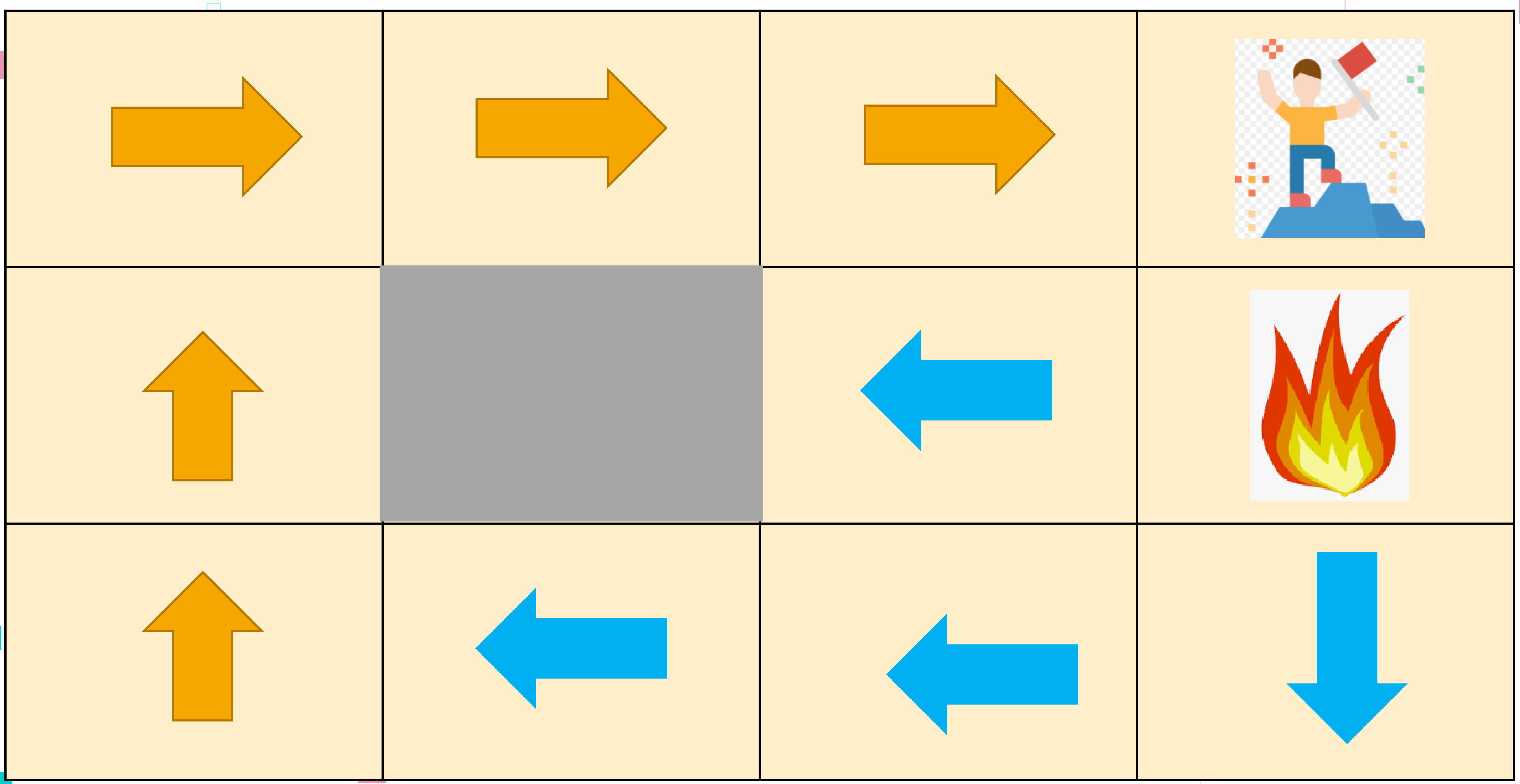
The Plan



The Policy- AI decide



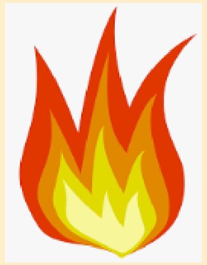
The Policy- AI decide based on MDP



$$V(s) = \max_a (\gamma \sum_{s'} P(s'|s,a)\,[R(s,a,s') + V(s')]$$

γ=0.9

Policy vs Plan

V = 0.37	**V = 0.52**	**V = 0.72**	**R= +1**
V = 0.27	▨	**V = 0.065**	 **R= -10**
V =	**V =**	**V =**	**V = 0.**

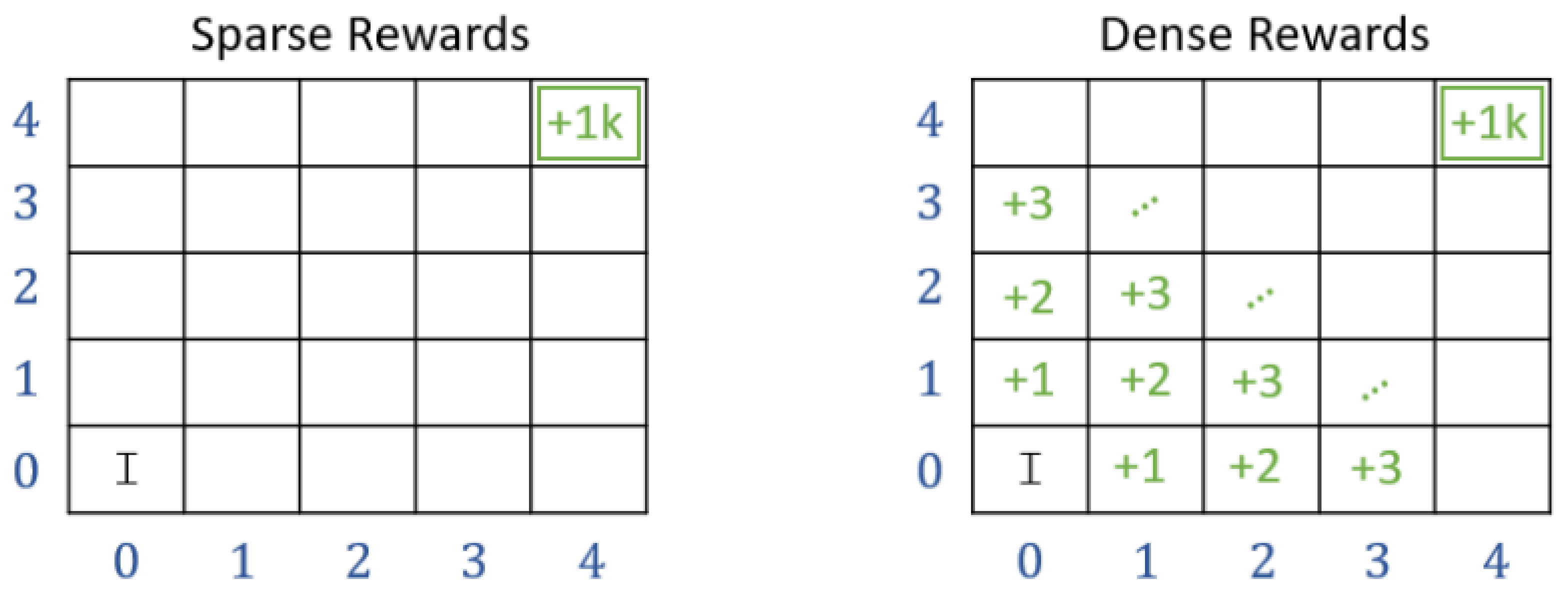
Rewards:

- Rewards are signals given to an agent to indicate the desirability of a particular state or action.
- Importance in RL: Rewards drive the learning process by providing feedback on the agent's actions.
- **Living Penalty Rewards**: A small negative reward given at each time step to encourage the agent to reach the goal quickly.
  - Prevents the agent from idling and encourages efficiency.
  - Example: In a gridworld, the agent might receive -0.01 reward for each move to incentivize reaching the goal state sooner.

Rewards:

- **Sparse Rewards**: rewards that are given infrequently, often only when significant milestones or goals are achieved.
  - Harder for the agent to learn since feedback is rare.
  - Example: In a maze, the agent receives a reward only when it reaches the exit.
  - Pick-and-Place: A robot arm must pick up an object and place it in a target location. Rewards are sparse, typically given only when the task is successfully completed.
- **Dense Rewards**: Rewards provided frequently, giving continuous feedback on the agent's actions.
  - Easier for the agent to learn as it receives more immediate feedback.
  - Example: In a game, the agent receives points for every correct move or action

Rewards:



http://forns.lmu.build/classes/spring-2020/cmsi-432/lecture-13-2.html

$$V(s) = \max_a (\gamma \sum_{s'} P(s'|s,a)\,[R(s,a,s') + V(s')]$$

γ=0.9

Living Penalty

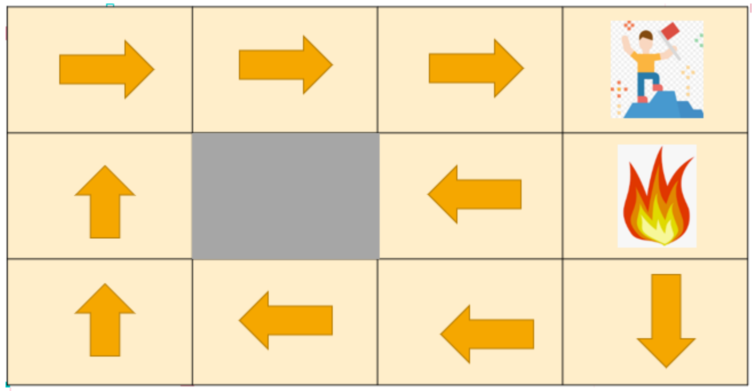
R = -0.03	**R = -0.03**	**R = -0.03**	**R= +1**
R = -0.03	▨	**R = -0.03**	**R= -1**
R = -0.03	**R = -0.03**	**R = -0.03**	**R = -0.03**

$$V(s) = \max_a(\gamma \sum\nolimits_{s'} P(s'|s,a)\,[R(s,a,s') + V(s')]$$

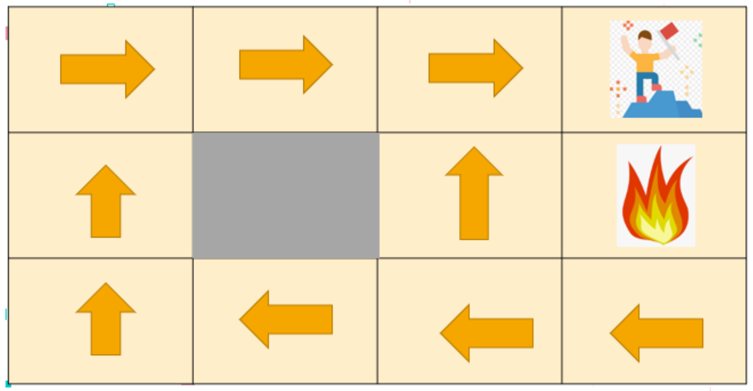
$$\gamma=0.9$$

Living Penalty

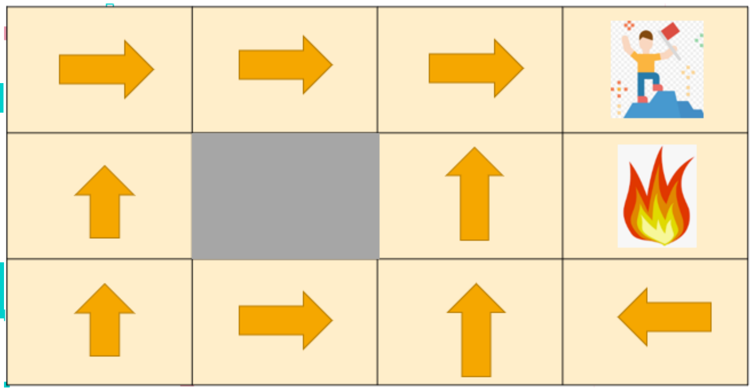
R(s)=0



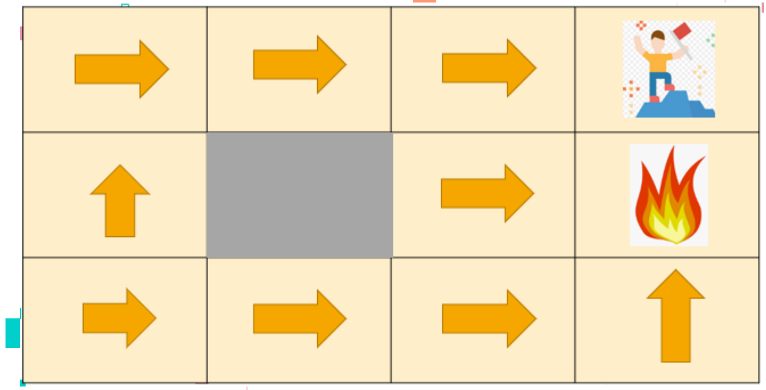
R(s)=-0.04



R(s)=-0.5



R(s)=-10.0



Comparison Rewards:

- Living Penalty: Encourages efficiency but can lead to suboptimal policies if not tuned correctly.
- Sparse Rewards: Difficult learning but can lead to robust policies if the agent succeeds.
- Dense Rewards: Easier learning but may require careful design to avoid unintended behaviors.

Q-learning

What is Q-Learning?

- A value-based reinforcement learning algorithm used to find the optimal action-selection policy for any given finite Markov decision process (MDP).
- Q-value (Quality): Measures the value of an action in a particular statePolicy: Strategy that the agent employs to determine the next action based on the current state
- Goal is to maximize the Q-value to find the optimal action-selection policy.
- The Q table helps to find the best action and maximize the expected reward

Q-Learning Algorithm

1. Initialize Q-table: A table where we keep Q-values for each state-action pair.
2. For each episode:
   - Initialize state s
   - Repeat (for each step of the episode):
     i. Choose an action A based on the current state s using a policy (eg: epsilon-greedy)
     ii. Take action a, observe reward R, and next state s'
     iii. Update Q-value

$$\mathrm{Q}(s,a) = Q(s,a) + \alpha[R(s,a) + \gamma \max_{a'}(Q(s',a') - Q(s,a)]$$

     iv. Set state s to the new state s'
   - Until state S is terminal
3. End

Policy(π) in Q-Learning

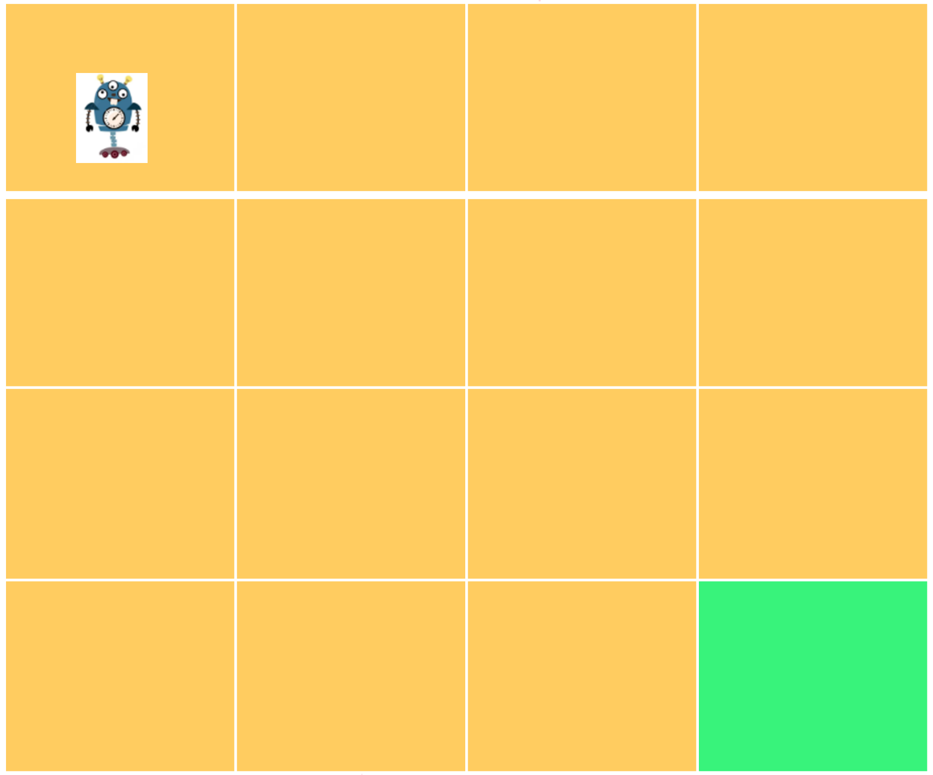
- A strategy used by the agent to decide the actions based on the current state
- Types of Policy:
  - ❖ Exploration vs. Exploitation:
    - Exploration: Trying new actions to discover their effects (e.g., taking random actions).
    - Exploitation: Using known information to maximize reward (e.g., selecting the action with the highest Q-value).
  - ❖ ϵ-Greedy Policy:
    - Exploration Rate (ϵ): Probability of choosing a random action.
    - Greedy Action Selection: With probability $1-\epsilon$, choose the action with the highest Q-value.

$$\pi(s) = \begin{cases} \textit{random action}, & \textit{with probability } \epsilon \\ \max\limits_{a}(Q(s,a)), & \textit{with probability } 1-\epsilon \end{cases}$$

    - Adaptive ϵ:Decay Strategy, Gradually reduce ϵ over time to shift from exploration to exploitation (High ϵ, more random actions (exploration), Low ϵ, more greedy actions (exploitation)

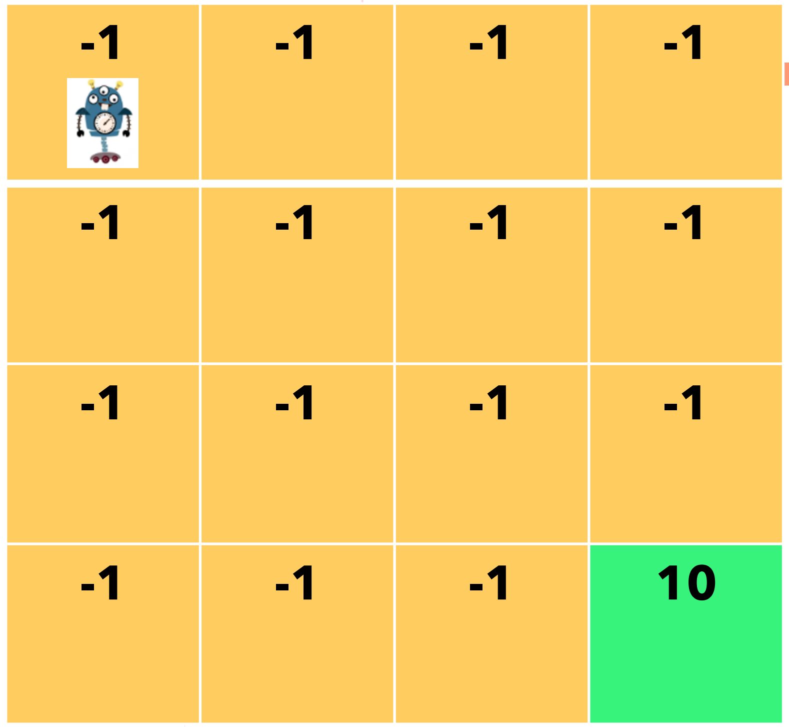
Example of ϵ-Greedy Policy in gridworld

- State: (0,0)
- Actions: $['up','down','left','right']$
- Q-values: $[-0.5, -0.2, -0.3, 0.0]$
- Policy Decision:
  - ❑With ϵ=0.1, 10% chance of random action.
  - ❑90% chance of choosing 'right' (action with Q-value 0.0).



Q-Learning Example Gridworld

- Environment: A 4x4 grid where the goal is to reach the cell (3,3)
- Reward of 10 for reaching the goal, -1 for each step otherwise
- Actions: Up, Down, Left, Right
- Learning Rate=0.1, Discount Factor= 0.9
- Q-Table Initialization: Initialize Q-values for all state-action pairs to 0



-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	10

Q-Learning Example Gridworld

Example at state (0,0), Action: down (exploration).

Old Q-value: -0.29 (in down matrix)

- Reward: -1
- Next State: (1, 0),
- Best Next Action: right, Q-value: -0.21,
- New Q-value calculation:

$$Q(s,a) = Q(s,a) + \alpha[R(s,a) + \gamma \max_{a'}(Q(s',a') - Q(s,a)]$$
$$Q(s,a) = -0.29 + 0.1[(-1) + 0.9(-0.21) - (-0.29)]$$
$$Q(s,a) = -0.38$$

- New Q-value = -0.38 of (0,0) down matrix

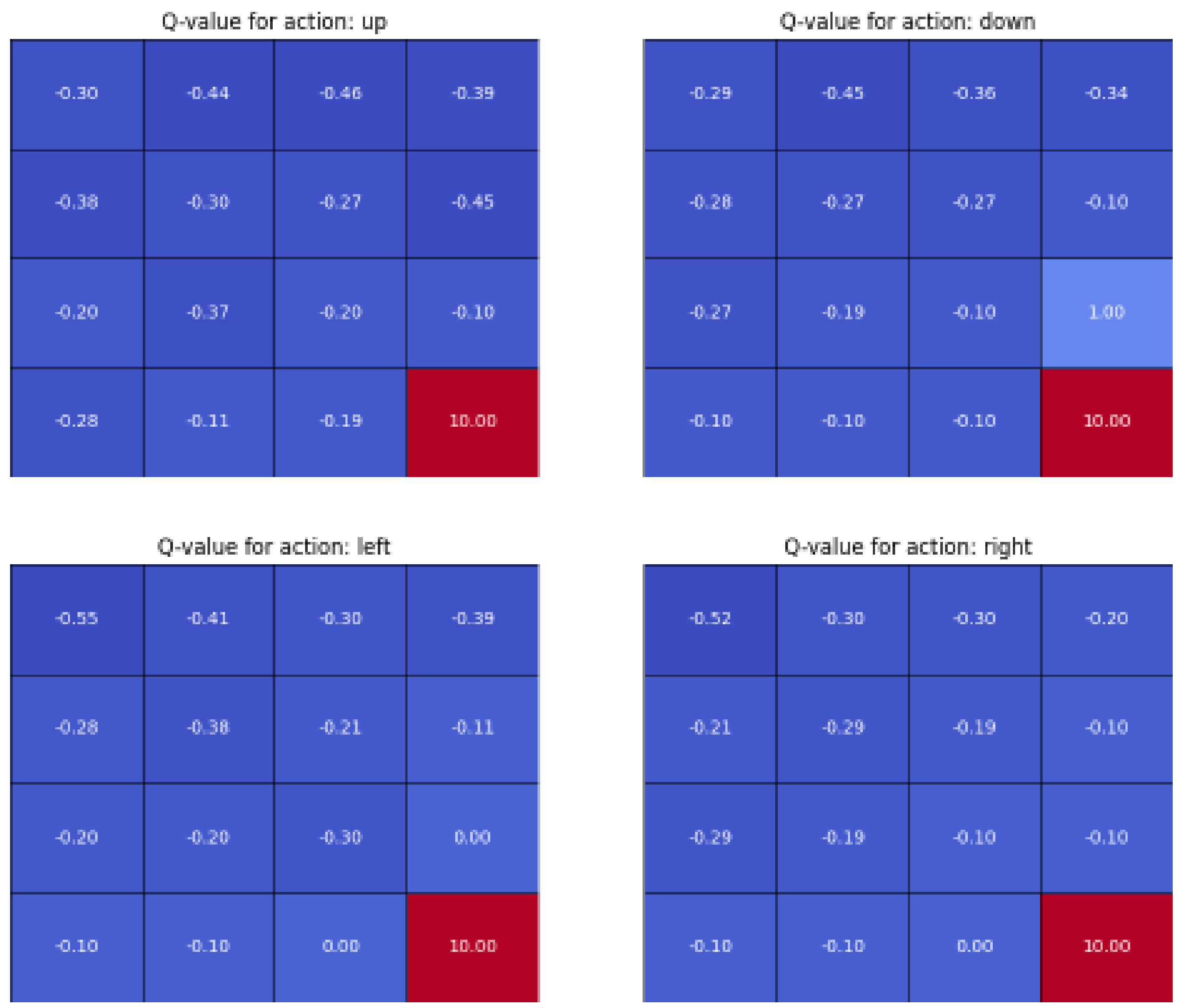
Now state (1,0), Action: right (exploitation).

Old Q-value: -0.21 (in right matrix)

- Reward: -1
- Next State: (1,1),
- Best Next Action: down, Q-value: -0.27,
- New Q-value calculation:

$$Q(s,a) = -0.21 + 0.1[(-1) + 0.9(-0.27) - (-0.21)]$$
$$Q(s,a) = -0.31$$

- New Q-value = -0.31 of (1,0) of right matrix



Q-value for action: up

-0.30	-0.44	-0.46	-0.39
-0.38	-0.30	-0.27	-0.45
-0.20	-0.37	-0.20	-0.10
-0.28	-0.11	-0.19	10.00

Q-value for action: down

-0.29	-0.45	-0.36	-0.34
-0.28	-0.27	-0.27	-0.10
-0.27	-0.19	-0.10	1.00
-0.10	-0.10	-0.10	10.00

Q-value for action: left

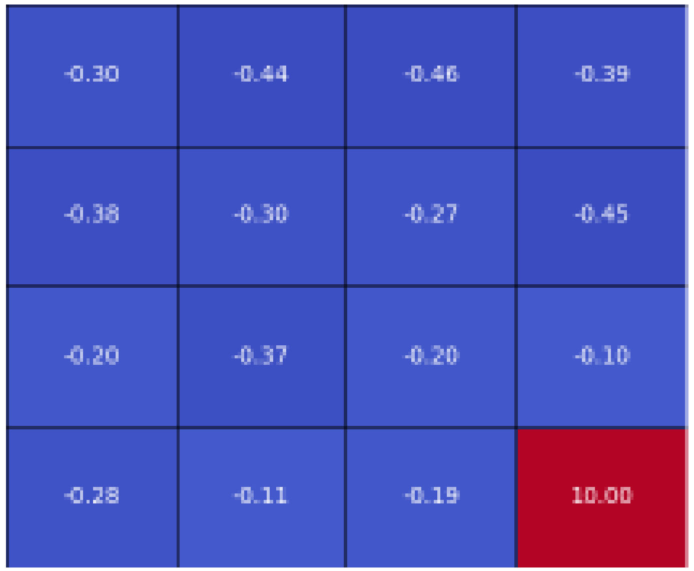
-0.55	-0.41	-0.30	-0.39
-0.28	-0.38	-0.21	-0.11
-0.20	-0.20	-0.30	0.00
-0.10	-0.10	0.00	10.00

Q-value for action: right

-0.52	-0.30	-0.30	-0.20
-0.21	-0.29	-0.19	-0.10
-0.29	-0.19	-0.10	-0.10
-0.10	-0.10	0.00	10.00

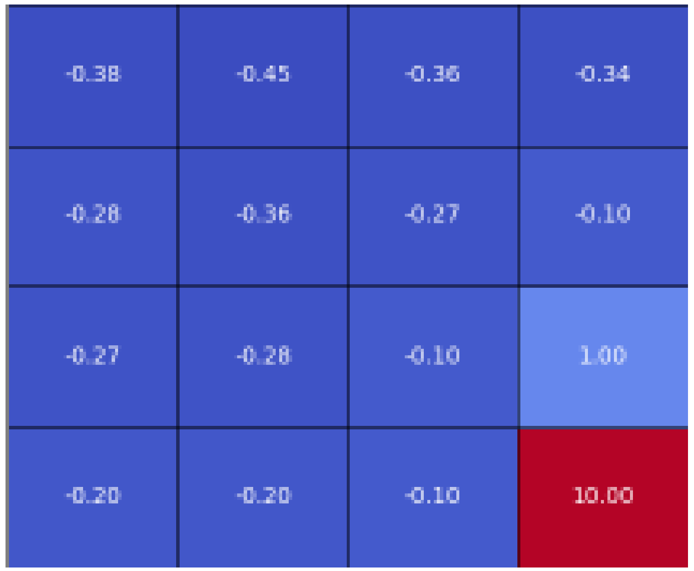
Q-table after episode 2

Q-value for action: up



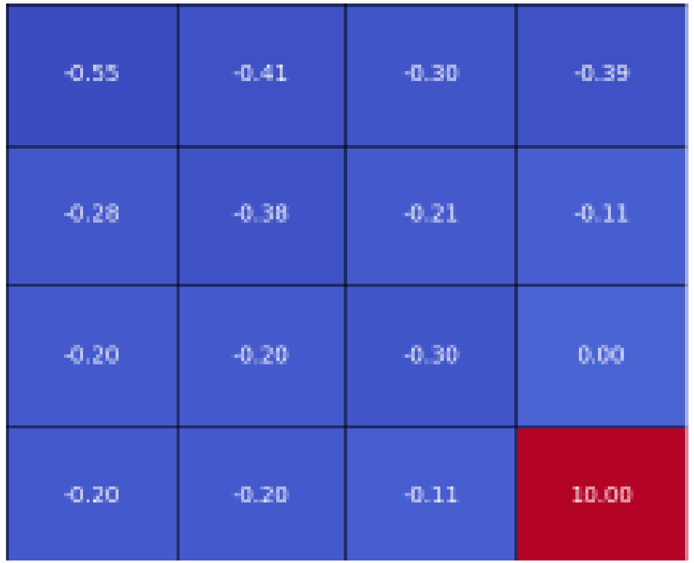
-0.30	-0.44	-0.46	-0.39
-0.38	-0.30	-0.27	-0.45
-0.20	-0.37	-0.20	-0.10
-0.28	-0.11	-0.19	10.00

Q-value for action: down



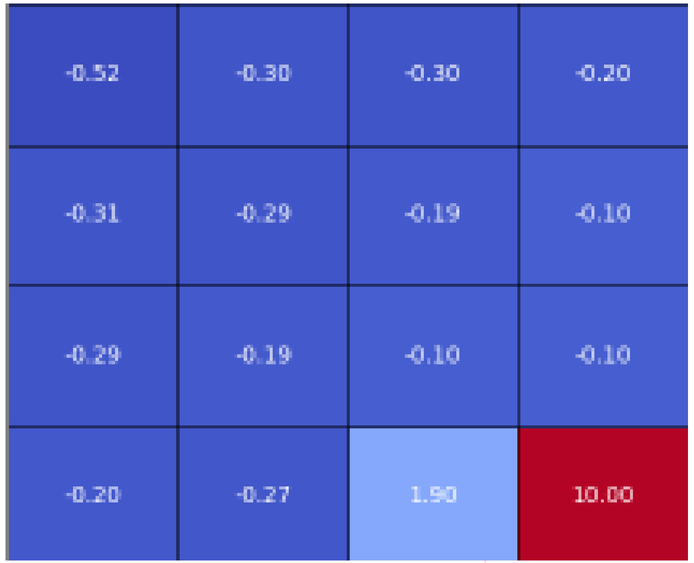
-0.38	-0.45	-0.36	-0.34
-0.28	-0.36	-0.27	-0.10
-0.27	-0.28	-0.10	1.00
-0.20	-0.20	-0.10	10.00

Q-value for action: left



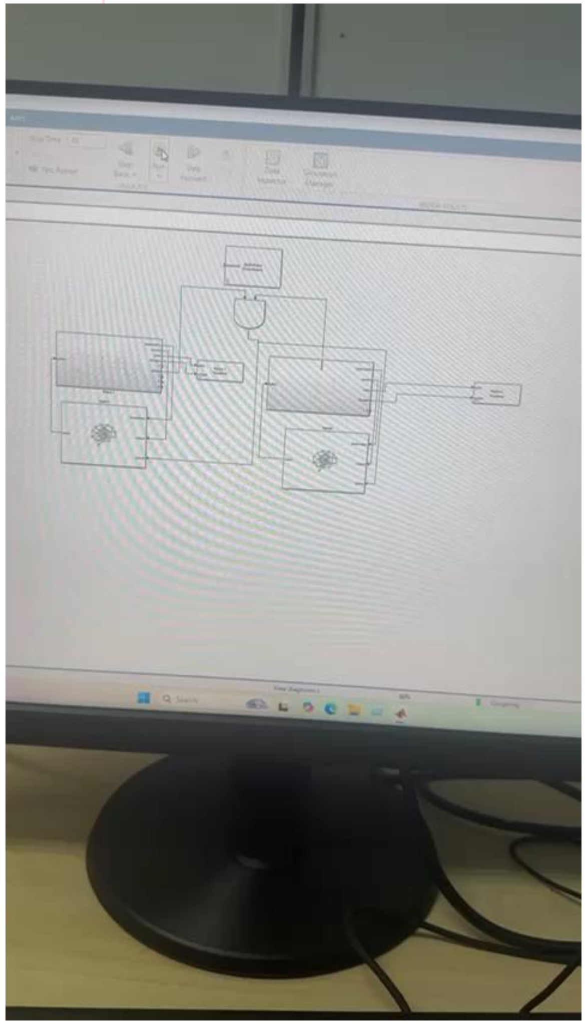
-0.55	-0.41	-0.30	-0.39
-0.28	-0.38	-0.21	-0.11
-0.20	-0.20	-0.30	0.00
-0.20	-0.20	-0.11	10.00

Q-value for action: right

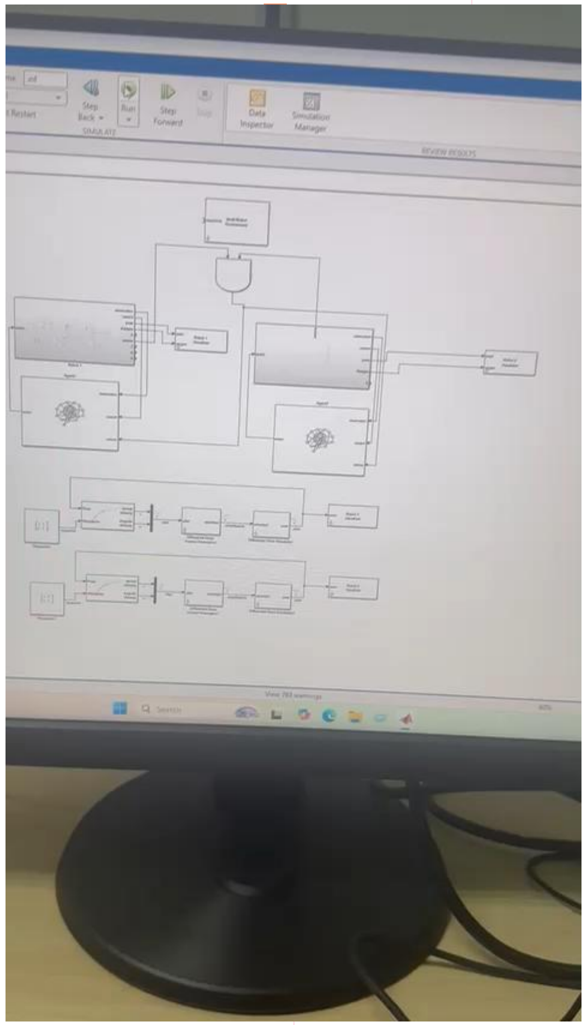


-0.52	-0.30	-0.30	-0.20
-0.31	-0.29	-0.19	-0.10
-0.29	-0.19	-0.10	-0.10
-0.20	-0.27	1.90	10.00

Some of DRL Agents in Action!

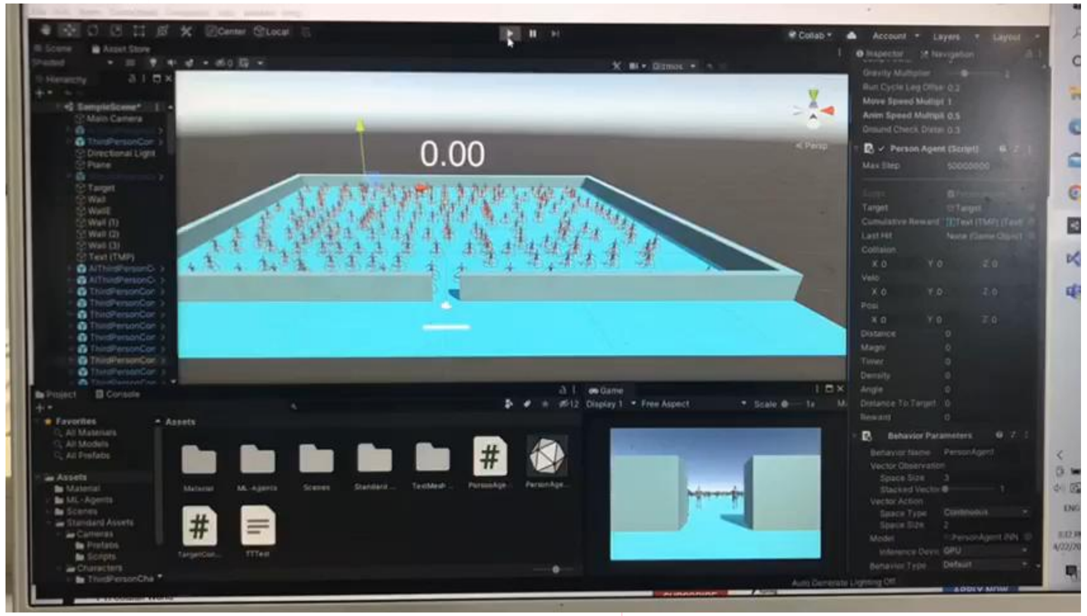


Unconverged Model



Converged Model

Some of DRL Agents in Action!



Thank You

&

Enjoy AI