# OpenAI Gym

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, Wojciech Zaremba, OpenAI

João Gabriel Corrêa Krüger
Paulo José Machado Filho

# Gym - What is it?

- Python environment for Reinforcement Learning
- Contains different settings (environments) spanning through text, games and computational problems

# Motivation - RL

- Very general, encompasses all the problems that involves making a series of decisions
  - Controlling a robot's motors
  - Making business decisions
  - Video games
- Algorithms that started presenting good results in many problems
  - DeepMind Atari
  - AlphaZero (formerly AlphaGo and AlphaChess)

# Motivation - Gym

- The need for better benchmarks
  - Lack of a big repository for training RL algorithms, like ImageNet is for supervised learning
- Lack of standardization of environments in publications
  - Makes it harder to replicate results

# Gym - Goals

- Have many environments with a common interface
- Scoreboard
  - Inspired by Kaggle
- Convenience
- Serve as the “ImageNet” of reinforcement learning

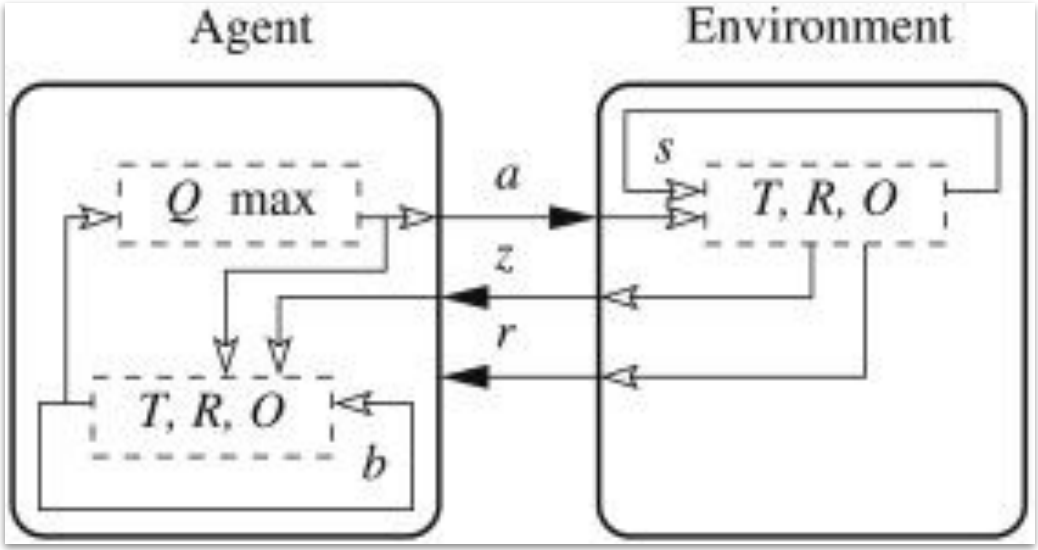
# Concepts

What is necessary to know before getting into Gym

# POMDP

- Partially observable Markov decision process (MDP)
- Generalization of the Markov process
- Assumes there is an underlying MDP for the problem, but the current state is unknown
- Keeps probabilities for the possible current states

# POMDP




Littman, 2009

# Agent

- Interact with the environment
- Goes back to classical AI
  - There are entire books about agents
- Many different types of agents
  - Simple reflex,  goal, model
  - RL focuses mostly on learning agents
- Many strategies about how to approach same problem

# Action

- How the agent interacts with the environment
- Given a certain state, there is a set of actions where the agent can choose from
- Particular to a given problem or state

# Observation

- The perception of the agent about its environment

- Changes with each action and state

# Reward

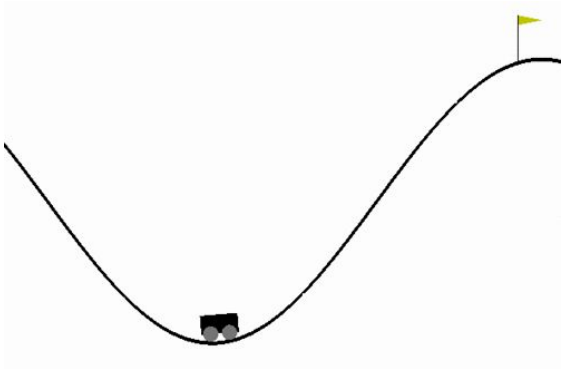
- Given after a certain favorable outcome happens
  - Can be the end of its task or after each successful action
- The learning agent tries to mold his actions based on his perception of “good” and “bad”
- The agent’s goal is to maximize his reward

# How does it all apply to Gym?

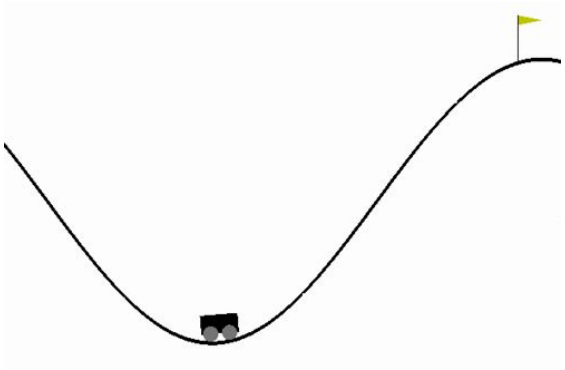
What are the main components of Gym?

# Episodes

- Basically everything that happens between the start state and the end state
- Rules can determinate the end of the episode



# Observation Space



- The observation space is what the Agent can see about the environment.

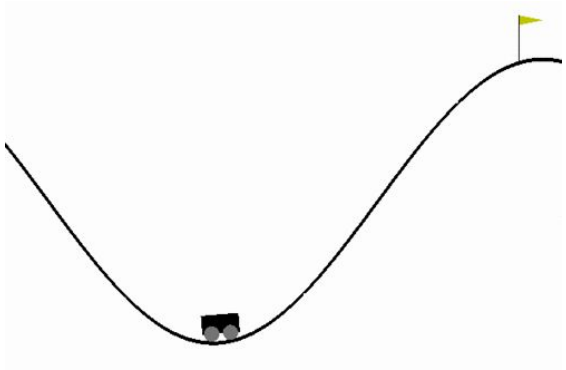
### Observation

Type: Box(2)

| Num | Observation | Min | Max |
|---|---|---|---|
| 0 | position | -1.2 | 0.6 |
| 1 | velocity | -0.07 | 0.07 |

# Action Space

- List of actions that the Agent can perform to interact with the environment



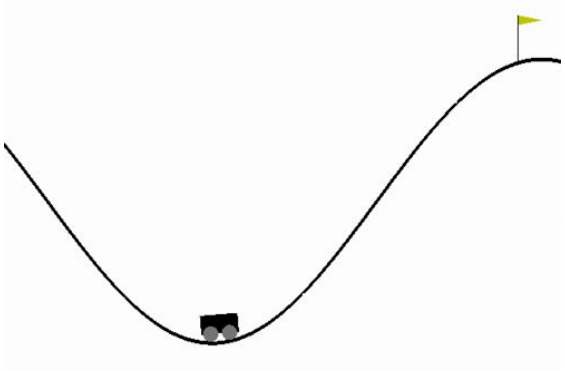
### Actions

Type: Discrete(3)

| Num | Action |
|---|---|
| 0 | push left |
| 1 | no push |
| 2 | push right |

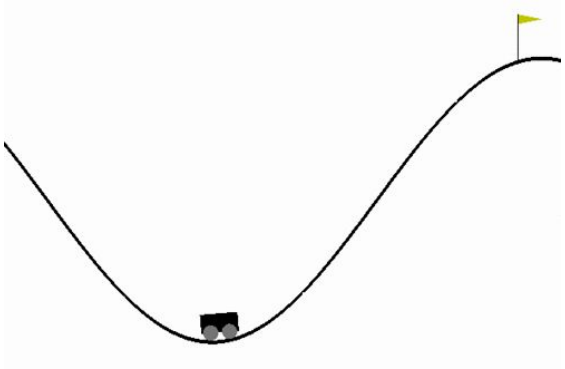
# Reward

- The Agent's objective is to maximize the reward
- Depends on the environment
- Example:
  - For the MountainCar Environment, the reward will only be positive when the car reaches the flag.



# Steps

- Step will receive an action and return what happened to the environment after that action was performed
- Input :
  - Action
- Output :
  - New State
  - Reward
  - Done
  - Info



# Design decisions

Why certain decisions were taken when developing Gym

# Design decisions

- Environments, not agents
- Emphasis on the final goal and how you got there
- Encourage peer review and not competition
- Versioning
- Monitor class and how it intends to be useful

# What Gym didn’t offer at the time of release

What wasn’t possible with Gym at the time of its release and what actions were taken to change that

# Multi-agent environment

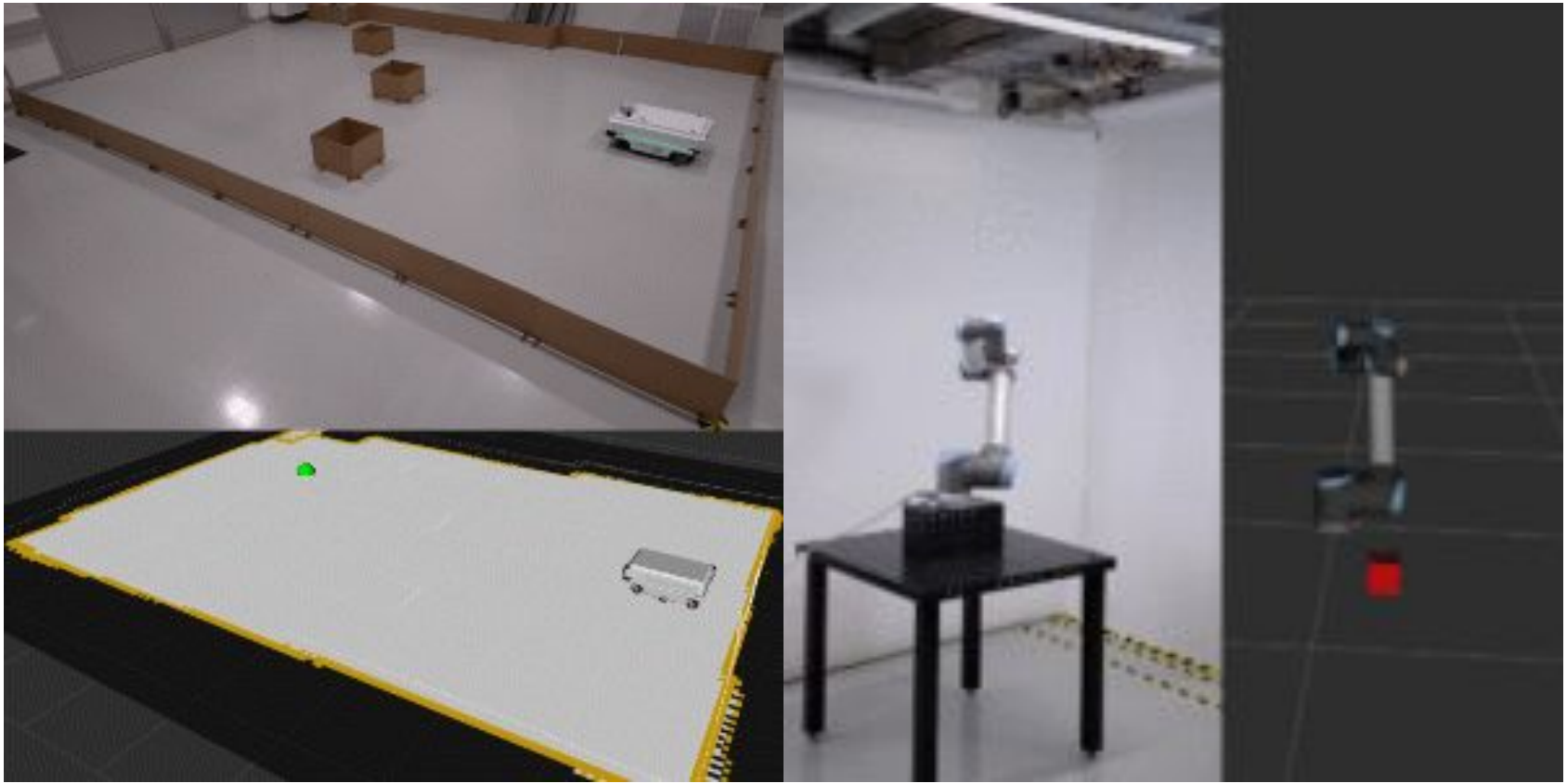
- It wasn’t possible at the time of release to use multiple agents
- There are third-party environments which support this
  - Pong
  - Predator and prey
  - Combat

# Curriculum and Transfer learning

- OpenAI didn’t release a set of related environments with increasing difficulty to test transfer learning in agents
  - They were very interested in this
- Can be done manually
  - Snake and Puck
    - Pygame

# Port to real world applications

- Hard due to the complexity of the simulations needed
- There have been progress in this area
  - robo-gym project
  - really complex



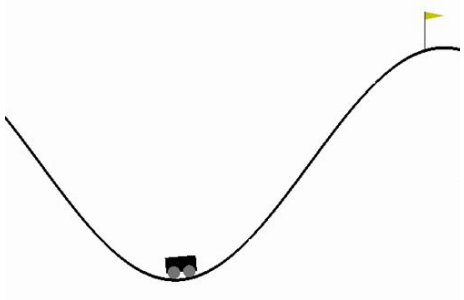
# Environments

What are the environments available on Gym?

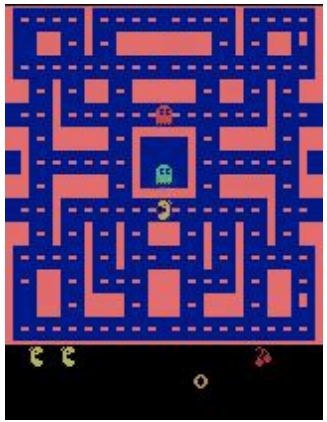
# Available Environments

- Based on Algorithms
- Classic problems in Reinforcement Learning
- Atari Games (RAM or Image)
- 2D or 3D robots
- Third Party Environments

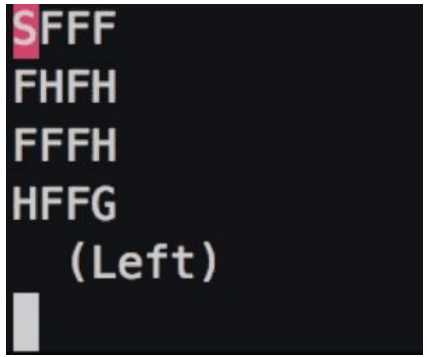
# Available Environments



MountainCar - v0



MsPacman - v0




FrozenLake - v0

# Demonstration

Coding some agents with Gym

- Random agent
- Simple-reflex agent
- Q-learning agent
- QNN agent

# Random agent

- At each step in the simulation takes a random step
- Works for every type of problem

- Can’t get more simple than that
- It’s really bad
- No learning involved

# Simple reflex

- Works for every type of problem
- Has a set of hard-coded rules to follow

- Still pretty simple
- Still no learning involved

# Q-Learning

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left( \underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)}^{\text{temporal difference}}$$

$$\text{new value (temporal difference target)}$$

- Learns!
- Work in discrete environments
  - Continuous environments can be digitized
- Has a table of size (states, actions)
- Updates the table based on the potential rewards and reward earned when taking an specific action in a specific state
- Bad explorer (needs epsilon and needs many iterations to converge)

# Q-Learning

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\Big(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}}\Big)}^{\text{temporal difference}}$$
$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{\text{new value (temporal difference target)}}$$

```
[[ 0.26750532  0.19727722  0.13726376  0.17226106]
 [ 0.27827805 -0.5628402   0.16870303  0.16498256]
 [ 0.28342283  0.327377    0.14610927  0.22189021]
 [ 0.3422643  -0.5485669   0.14548059  0.19471477]
 [ 0.19509248  0.30680603 -0.55029863  0.01584964]
 [ 0.25301278  0.51449656 -0.487648    0.02524841]
 [-0.5611712   0.363392   -0.7132637   0.22385378]
 [-0.47228703 -0.47469977  0.5137429  -0.23766291]
 [ 0.17930335 -0.44084     0.19943431 -0.03068562]
 [ 0.34401506  0.08189224  0.22766615 -0.65916735]
 [ 0.35449386  0.38711044 -0.7133499   0.25105685]
 [ 0.5102551  -0.216744   -0.40151045 -0.18541992]
 [-0.18365175 -0.01074147  0.08404386 -0.40878376]
 [-0.27485153  0.3103863  -0.05946351  0.25180778]
 [ 0.3274436   0.38642135  0.29573065  0.27928847]
 [ 0.33766204  0.11269939 -0.01776522  0.25525898]]
```

# QN / QNN

- Learns!
- Work in discrete environments*
- Similar to the Q-learning algorithm
- Can have many optimizations
  - Experience replay
- Has a neural network instead of a table
  - Learns the weights in the perceptrons
- Bad explorer

# Sources


- OpenAI Gym Beta
- Russell, Stuart, and Peter Norvig. "Artificial intelligence: a modern approach." (2002).
- Littman, M. L. (2009). *A tutorial on partially observable Markov decision processes. Journal of Mathematical Psychology, 53(3), 119–125.* doi:10.1016/j.jmp.2009.01.005
- Asawa, Chaitanya et al. “Using Transfer Learning Between Games to Improve Deep Reinforcement Learning Performance.” (2017).
- [2007.02753] robo-gym -- An Open Source Toolkit for Distributed Deep Reinforcement Learning on Real and Simulated Robots
- jr-robotics/robo-gym: An open source toolkit for Distributed Deep Reinforcement Learning on real and simulated robots.
- Deep Reinforcement Learning Explained