# IART - Aquarium

Group 55
Afonso Sá - up201604605
Guilherme Pinheiro - up201703867
Jorge Soares - up201809216

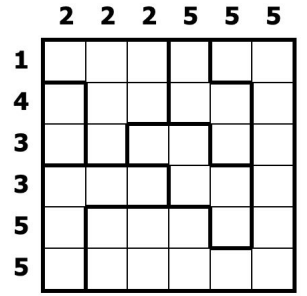
# Aquarium

The puzzle is played on a rectangular grid divided into blocks called "aquariums".

The players has to "fill" the aquariums with water up to a certain level or leave it empty.

The water level in each aquarium is one and the same across its full width.

The numbers outside the grid show the number of filled cells horizontally and vertically.




References:

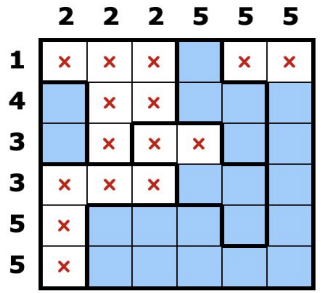
https://www.puzzle-aquarium.com/

https://www.pygame.org/docs/

https://gym.openai.com

Reinforcement Q-Learning

https://stable-baselines3.readthedocs.io/en/master/index.html



# What is reinforcement learning? Why do we need it?

Reinforcement learning (RL) is a machine learning technique that focuses on training an algorithm following the cut-and-try approach. The algorithm (agent) evaluates a current situation (state), takes an action, and receives feedback (reward) from the environment after each act.

On the whole, RL algorithms will learn and play the game. So, the main goal is to build an agent that is capable of solving the aquarium puzzle.

## Tools

- Programing language: Python
- Development environment: VS Code
- Libraries needed for development: Pygame, OpenAI Gym
- Library for PPO: https://stable-baselines3.readthedocs.io/en/master/index.html

# Environments

## Simple Reward

| Action | Reward |
|---|---|
| Finish | +50 |
| OverFlow | -50 |
| Diff between stateSOL and stateUser less than 0 | -10 |

## Complex Reward

| Action | Reward |
|---|---|
| Finish | +50 |
| OverFlow | -50 |
| Diff between stateSOL and stateUser less than 0 | -10 |
| Diff between stateSOL and stateUser equal to 0 | +10 |

| Aquarium Version | | | |
|---|---|---|---|
| | | Reward | |
| Board size | Number of aquariums | Simple | Complex |
| 3 | 2 | V1 | V6 |
| 3 | 3 | V5 | V2 |
| 4 | 2 | V3 | V4 |

# Environments Functions

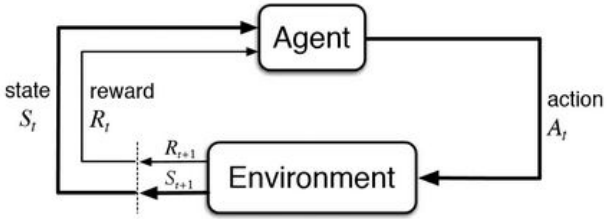
__init__()
reset()
step()
render()

**Action Space:** Number of aquariums (containers)

**Observation Space:** [W]-[EH]-[EV]

[W] -> water depth in each aquarium
[EH] -> deficit to the horizontal objectives
[EV] -> deficit to the vertical objectives

# Algorithms Implemented

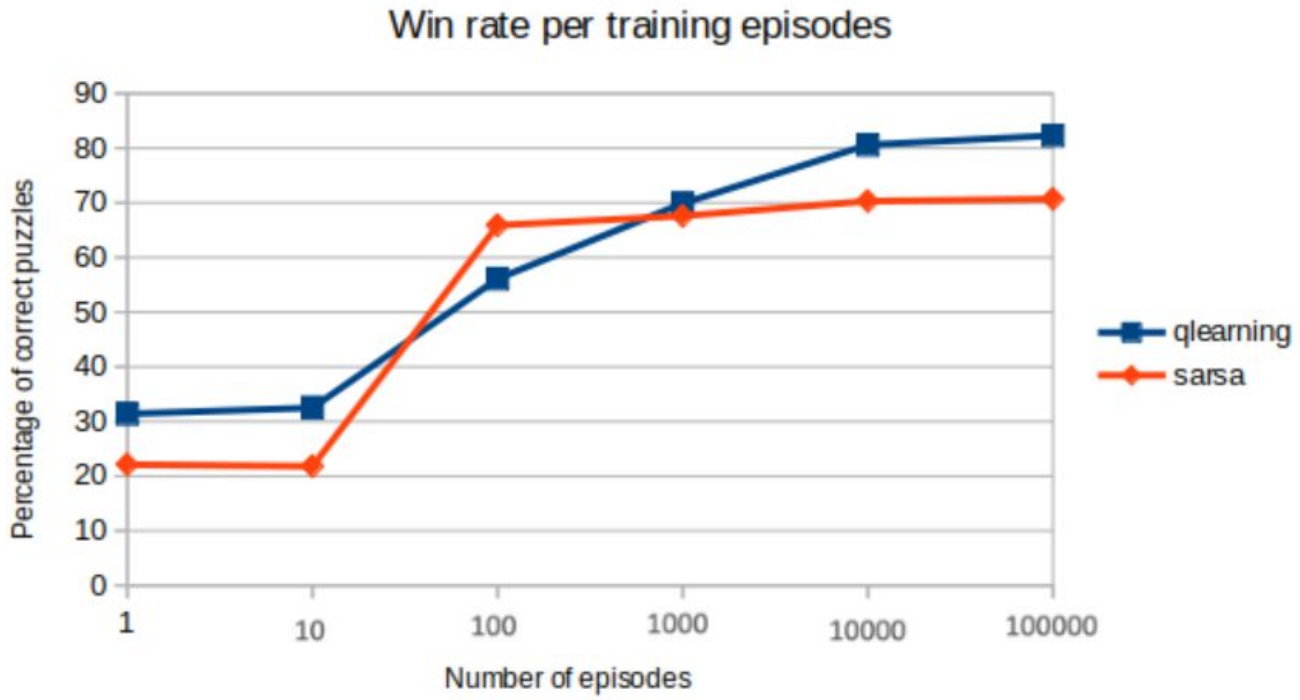



Algorithms we have implemented:

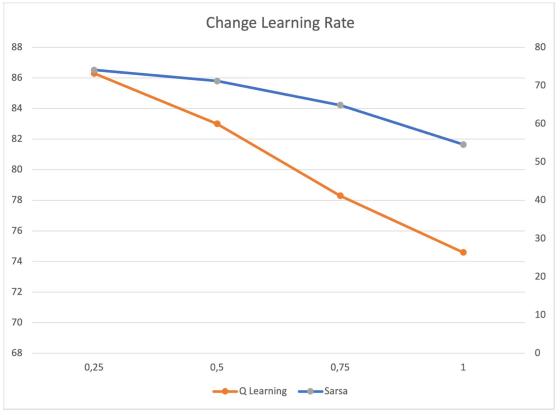
1. Q-Learning
2. SARSA

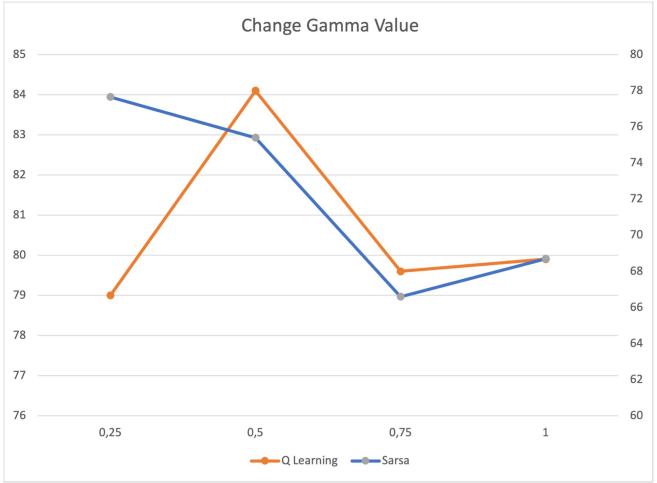
We tried to implement the PPO algorithm, using OpenAI's baselines library, but we found that it wasn't viable to keep using the library without some major refactoring to our code, thus we gave up on using this library and due to time constraints we couldn’t implement our own algorithm.

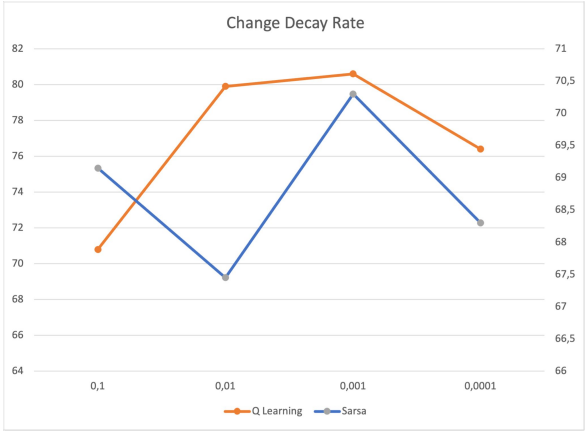
# Comparison between Q Learning and Sarsa



# Comparison between Q Learning and Sarsa



# Best HyperParameters and More Tests

For Q Learning:
* Number Episodes = 100000
* Learning Rate = 0.25
* Decay Rate = 0.001
* Gamma = 0.5

For Sarsa:
* Number Episodes = 100000
* Learning Rate = 0.25
* Decay Rate = 0.001
* Gamma = 0.25

We applied to the other different environments the same hyperparameters as before.

| Board size, # Aquariums | Simple | Complex |
|---|---|---|
| 3, 2 | 85.9% / 84.5% | 84.5% / 84.4% |
| 3, 3 | 41.0% / 34.0% | 45.7% / 42.2% |
| 4, 2 | 87.7% / 89.9% | 86.9% / 89.4% |

# Conclusions

The project was successfully implemented, having implemented the algorithms necessary to train an agent to solve the aquarium puzzle.

In order to make our algorithm solve the puzzle, due to its original complexity, we had to simplify the puzzle by reducing the size of the board and the number of containers(aquariums). This is due to one of the most limiting factors of RL, its time and computationally intensitivity to train an agent.

Based on all the previously collected data, we can take some conclusions about these two RL algorithms:

1. For both algorithms, if we increase the number of episodes, the test precision will be higher, this is to be expected, as the agent will have had more test boards to learn.
2. For both algorithms, if we increase the learning rate, the test precision will decrease.
3. For both algorithms, the best decay rate was 0.001, it was with that value that the algorithm had the best results.
4. For Q Learning, the value of gamma=0.5 is better. However, for Sarsa, it's preferable use gamma=0.25.