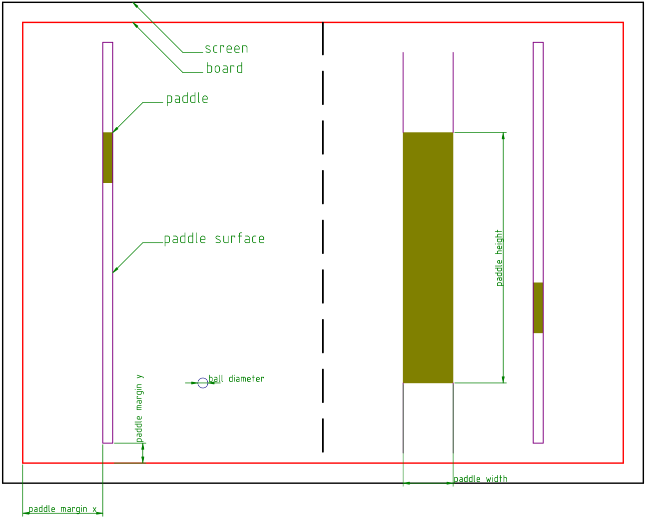
Implementation of the simulator

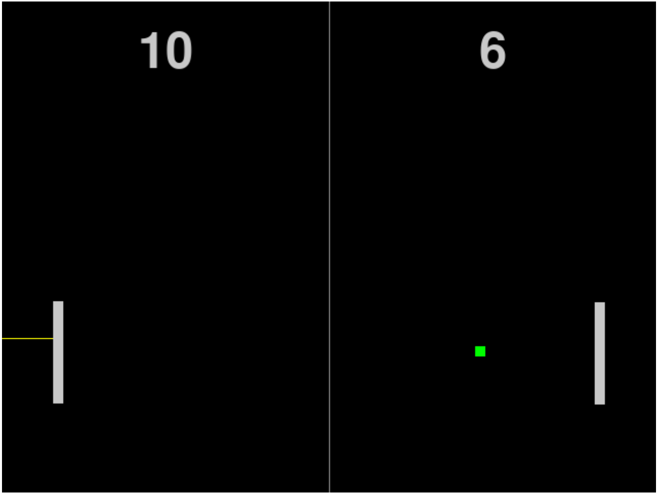
- The Pong is implemented using `pygame`
- Two modes of operation:
  - **Training**: no drawing, only physics (machine speed).
  - **Play**: Visually see the trained controller (human speed).
- Each controller is trained against a robotic reference controller that only follows the ball.
- Same CPU time for all controllers, by default 30 min.

Blueprints



screen
board
paddle
paddle surface
ball diameter
paddle margin y
paddle height
paddle width
paddle margin x

Pygame implementation



State mapping (QL and SARSA)

- How can we translate the state of the game $\theta = (v_1, v_2, \ldots, v_n)$ (position and velocity of the ball, paddles...) into a single number, the state $0 \leq s < N$?

$$\theta \xrightarrow{?} s$$

- We first **discretize** all the variables $\tilde{\theta}$, then we assign a number to the ordered values of $\tilde{\theta}$.
- The number of states N grows **exponentially** with the number of variables n.

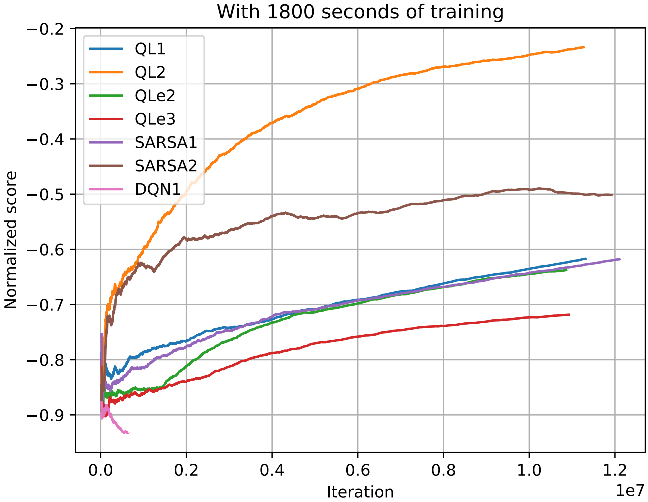
Measure of training performance

- In order to measure how well a controller is playing against the opponent, we use a normalized score d. Let S_C be the score of the controller and S_O of the opponent,

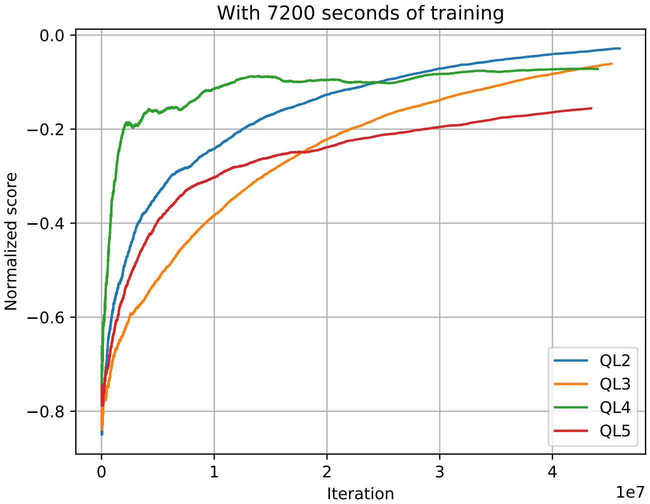
$$d = 2\frac{S_C - S_O}{S_C + S_O}$$

- The value d is less, equal or bigger than 0 if the controller behaves worse, similarly or better than the opponent, respectively.
- We want to maximize the d value.

Results: 30 min of training



Results: 2h of training



Thanks for your attention.

- R. S. Sutton and A. G. Barto *Reinforcement Learning: An Introduction* (Very good introduction to RL, draft available online)
- V. Mnih, K. Kavukcuoglu, D. Silver et al. *Human-level control through deep reinforcement learning* Nature 2015 – 10.1038/nature14236 (Other Atari games)