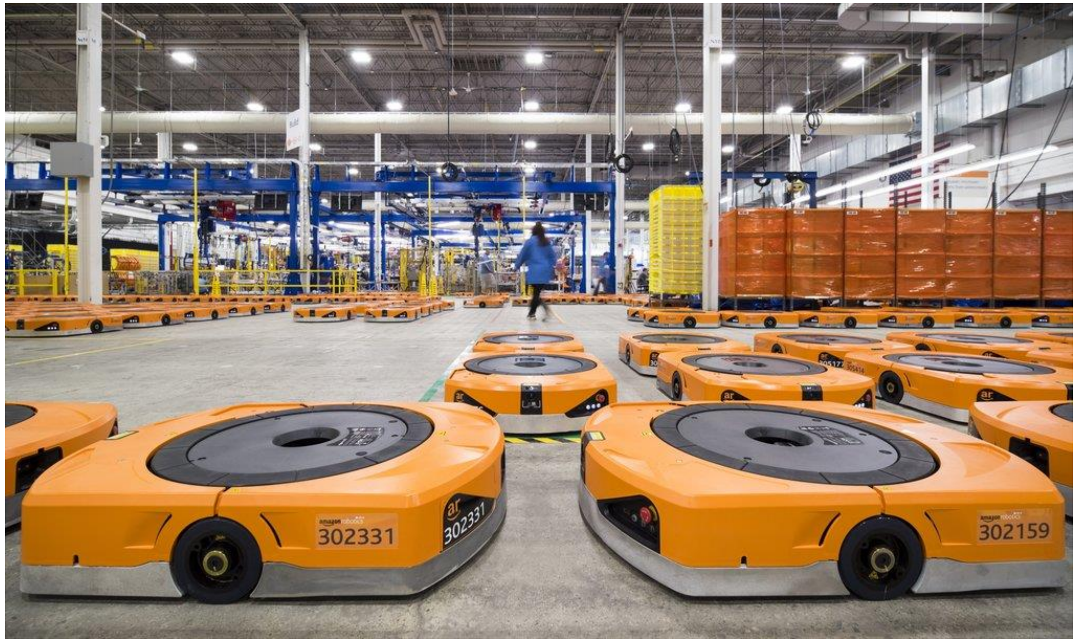
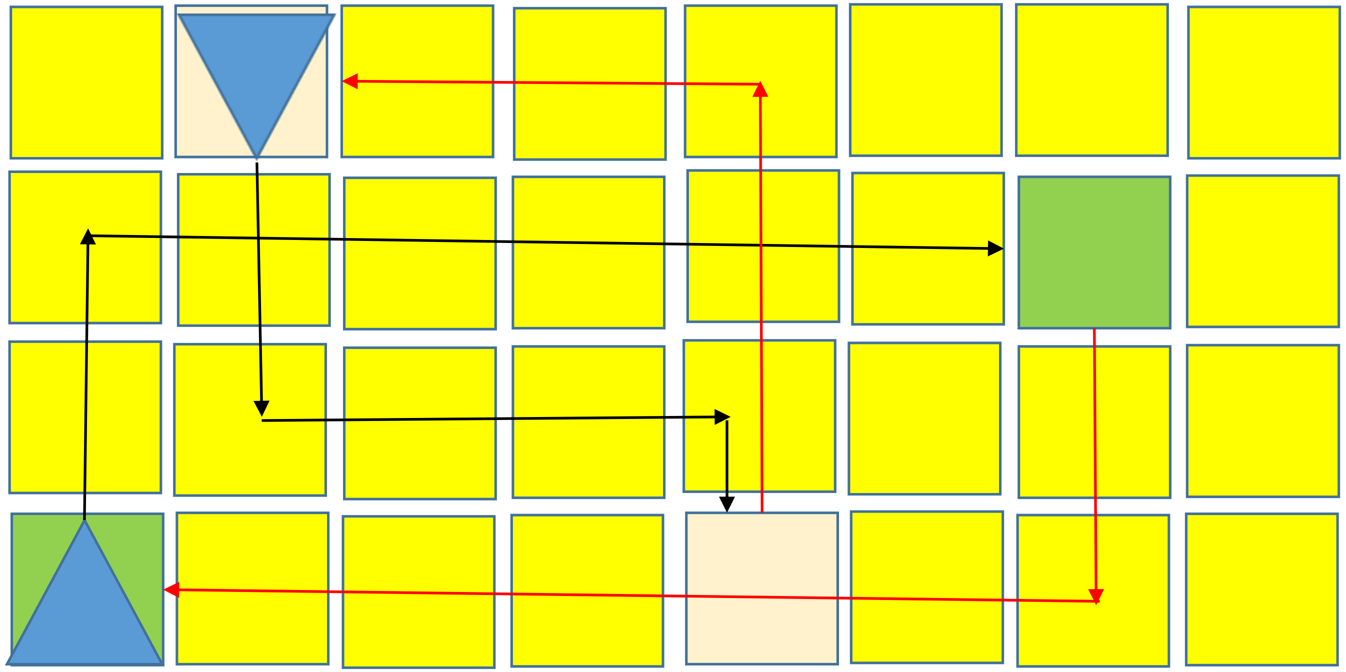
Robot-Walker

A simple example of a neural network development



Paolo Poli and Alessandro Zanetti

The goal



Each robot has to walk between the two assigned bases avoiding crashes with other robots

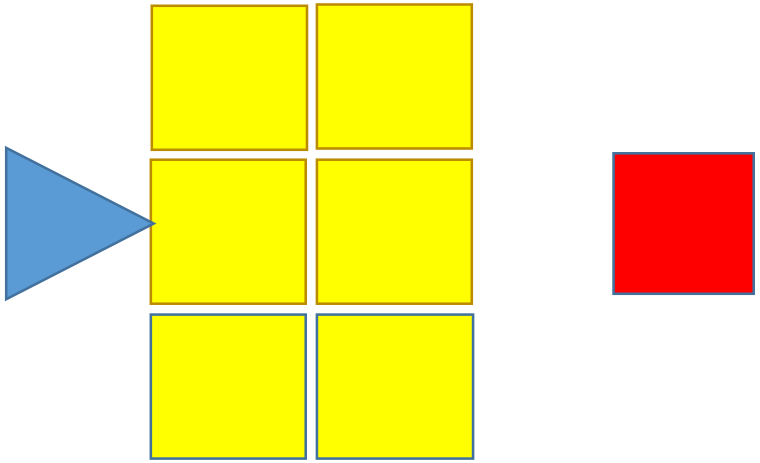
Programs and languages

- Python to develop the neural network

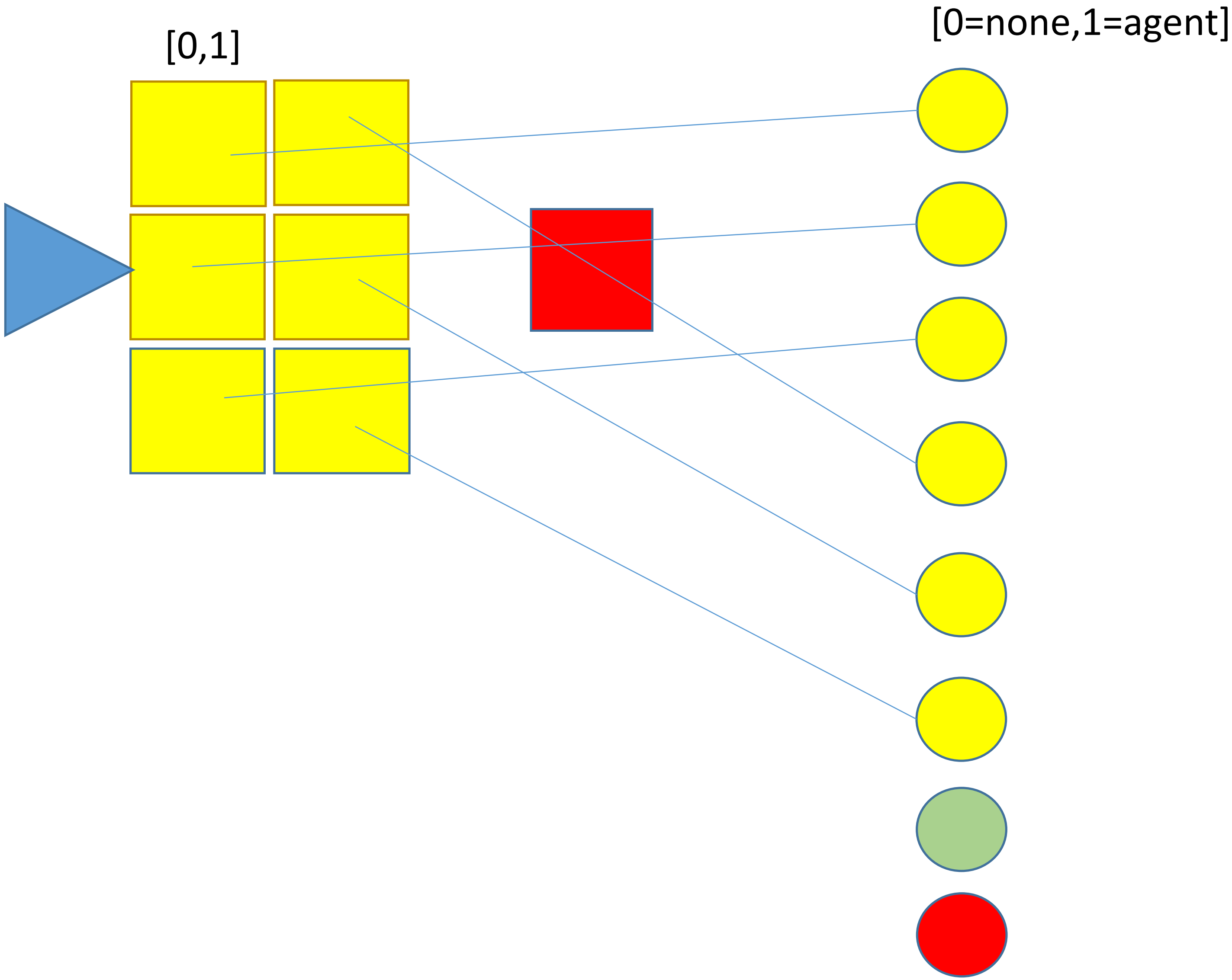
Programs and languages

- Python to develop the neural network
- Unity 3D to show the dynamic of the simulation

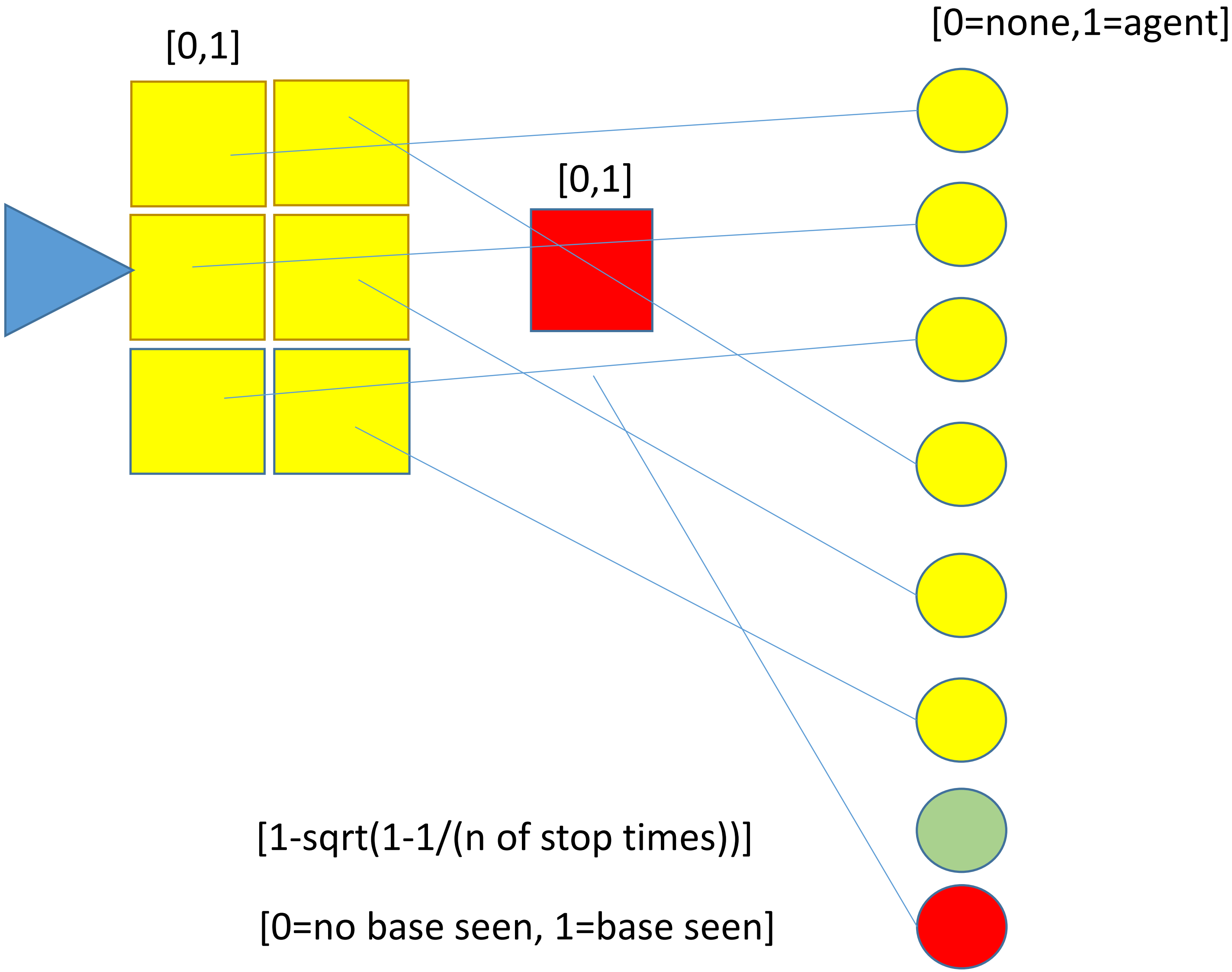
Neural Network



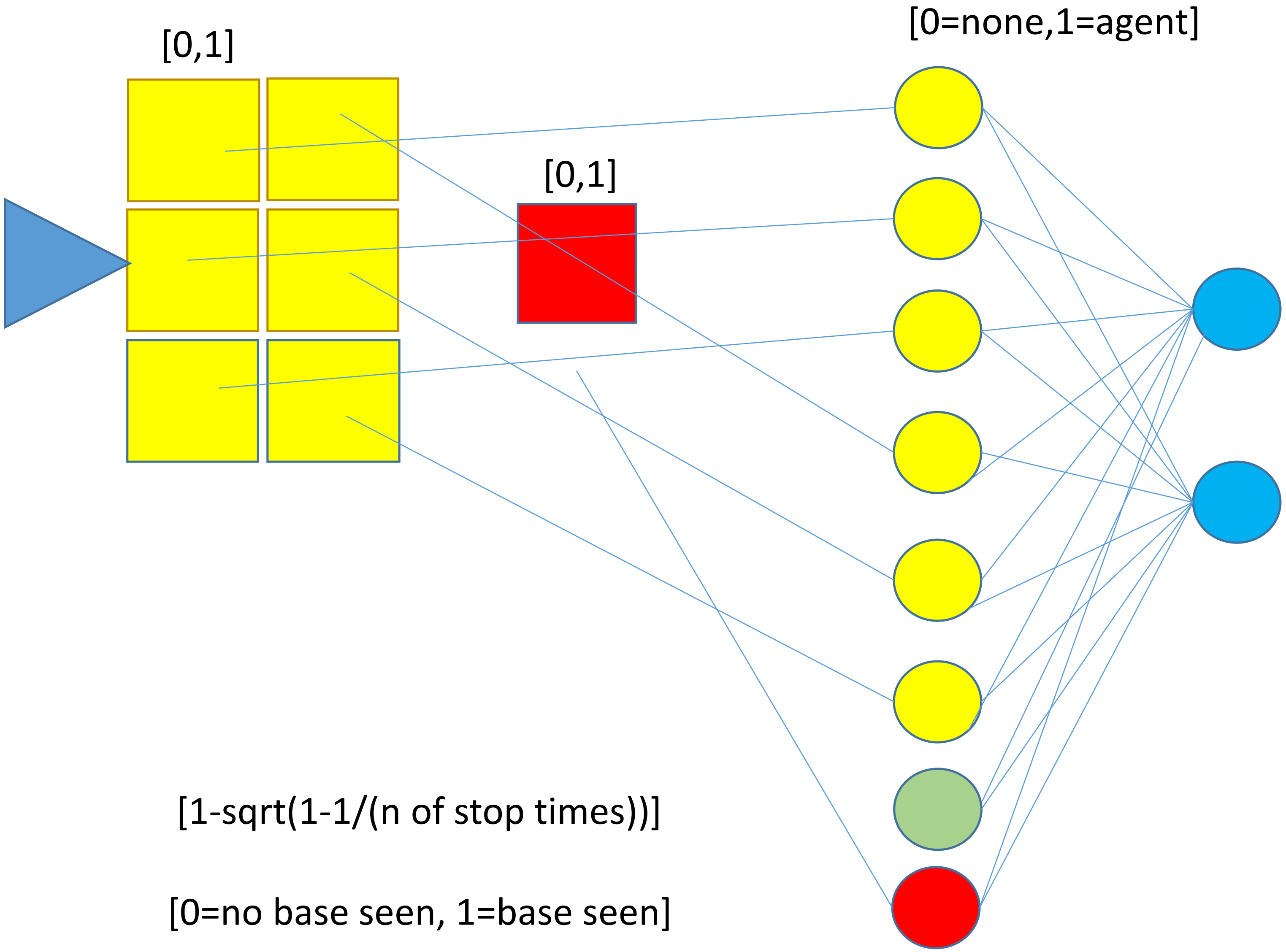
Neural Network



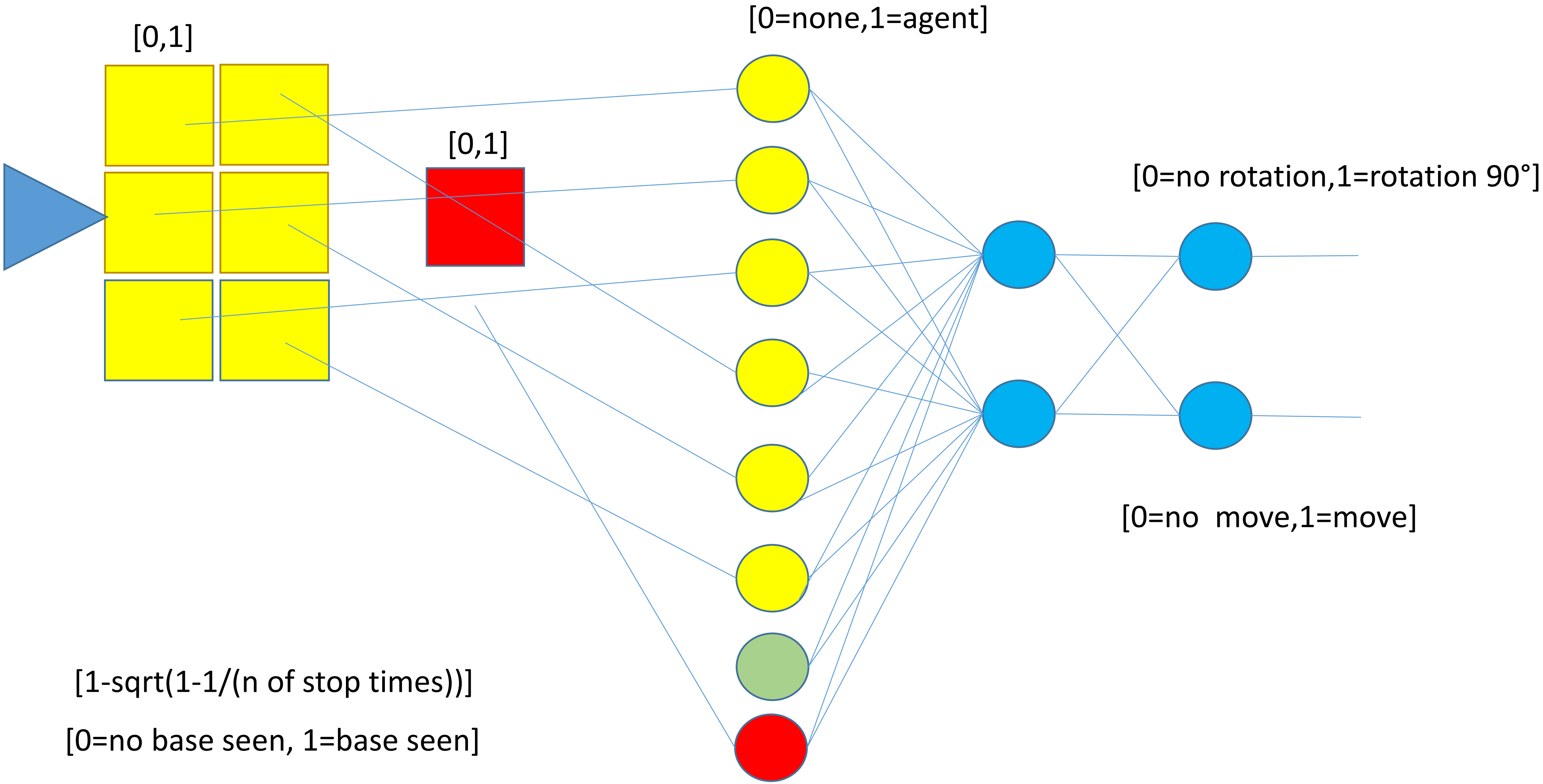
Neural Network


[0,1]
[0,1]
[0=none,1=agent]
[1-sqrt(1-1/(n of stop times))]
[0=no base seen, 1=base seen]

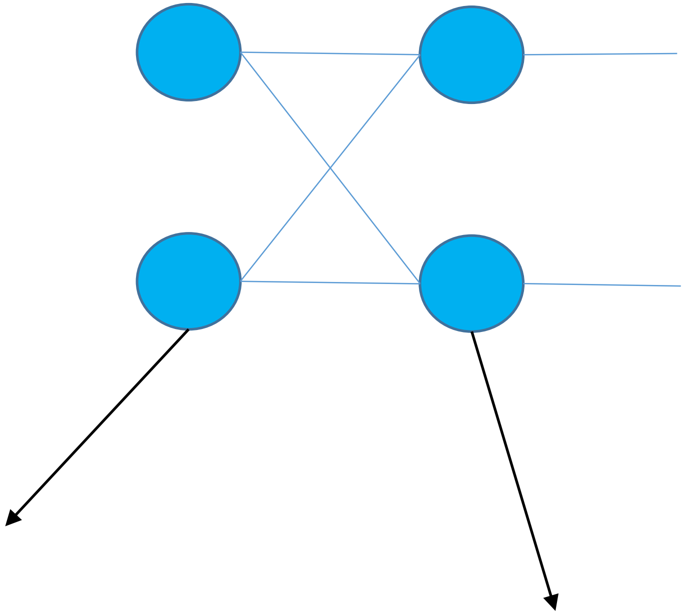
Neural Network


[0,1]
[0,1]
[0=none,1=agent]
[1-sqrt(1-1/(n of stop times))]
[0=no base seen, 1=base seen]

Neural Network


[0,1]
[0,1]
[0=none,1=agent]
[0=no rotation,1=rotation 90°]
[0=no move,1=move]
[1-sqrt(1-1/(n of stop times))]
[0=no base seen, 1=base seen]

Activation functions



-1 if $\sum wx \leq 0$

1 if $\sum wx > 0$

0 if $\sum wx \leq 0$

1 if $\sum wx > 0$

Total number of parameters=16+4=20

Neural network training

- Greedy method (it’s not compulsory to use specific methods in ML literature)

Neural network training

- Greedy method (it’s not compulsory to use specific methods in ML literature)

- Two random matrix were used, at the begin of the simulation with value between -1 and 1

Neural network training

- Greedy method (it’s not compulsory to use specific methods in ML literature)

- Two random matrix were used, at the begin of the simulation with value between -1 and 1

- The training works increasing any value of the matrix of a random value between –ω and ω

Neural network training

- Greedy method (it’s not compulsory to use specific methods in ML literature)
- Two random matrix were used, at the begin of the simulation with value between -1 and 1
- The training works increasing any value of the matrix of a random value between –ω and ω
- The best performed matrix was chosen, any time the result was better of previously results

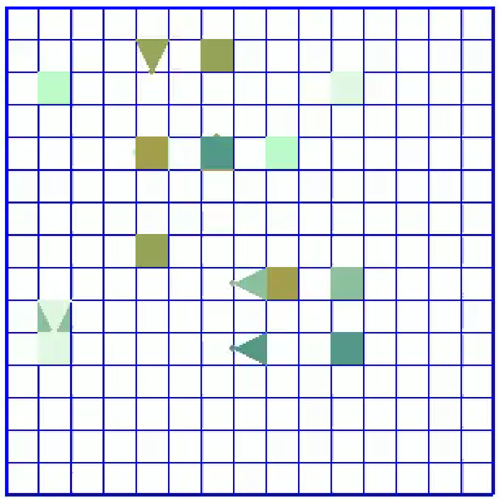
Cost function

The cost function chosen, is the sum of the number times each robot reaches one of the two assigned bases

Results

The cost function chosen, is the sum of the number times of each robot reaches one of the two assigned bases

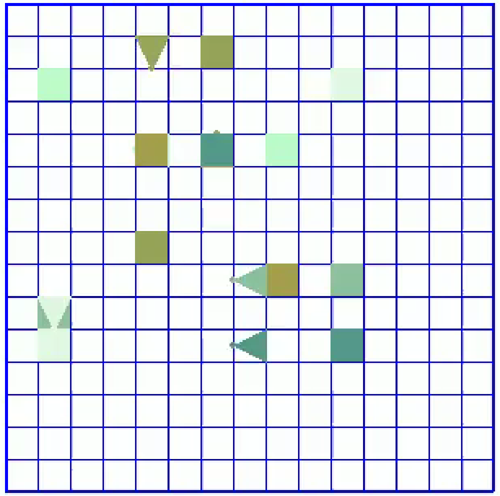
Untrained



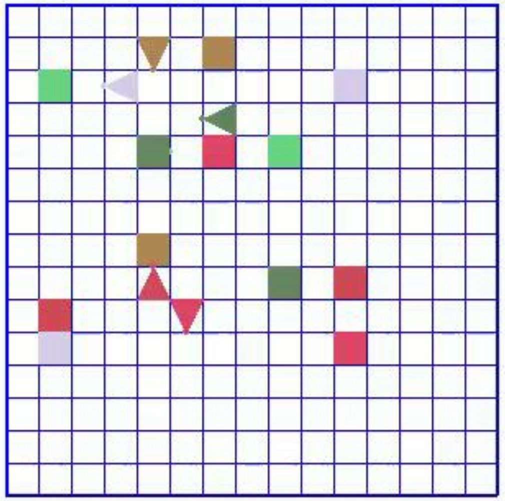
Results

The cost function chosen, is the sum of the number times of each robot reaches one of the two assigned bases

Untrained

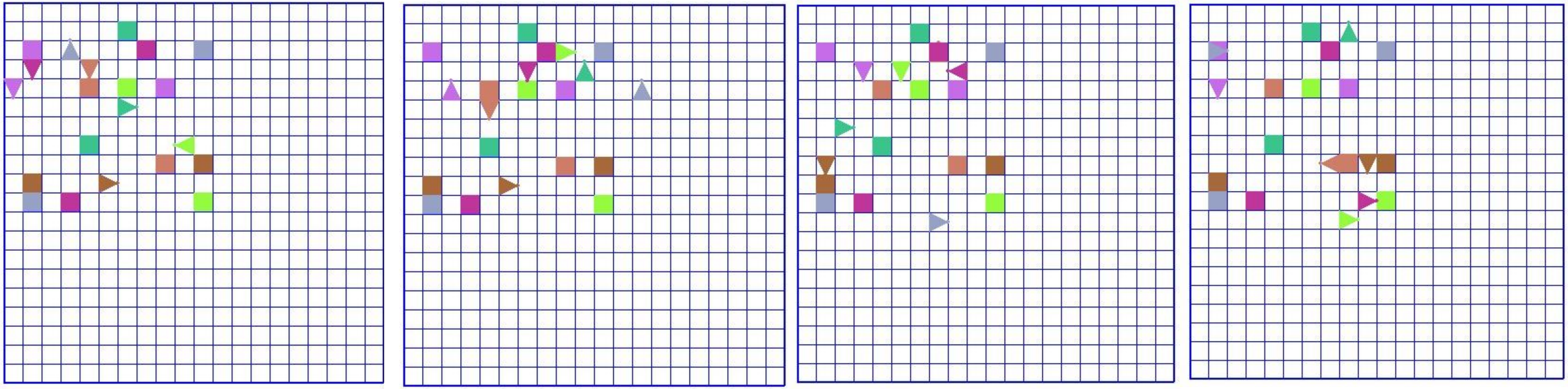


trained

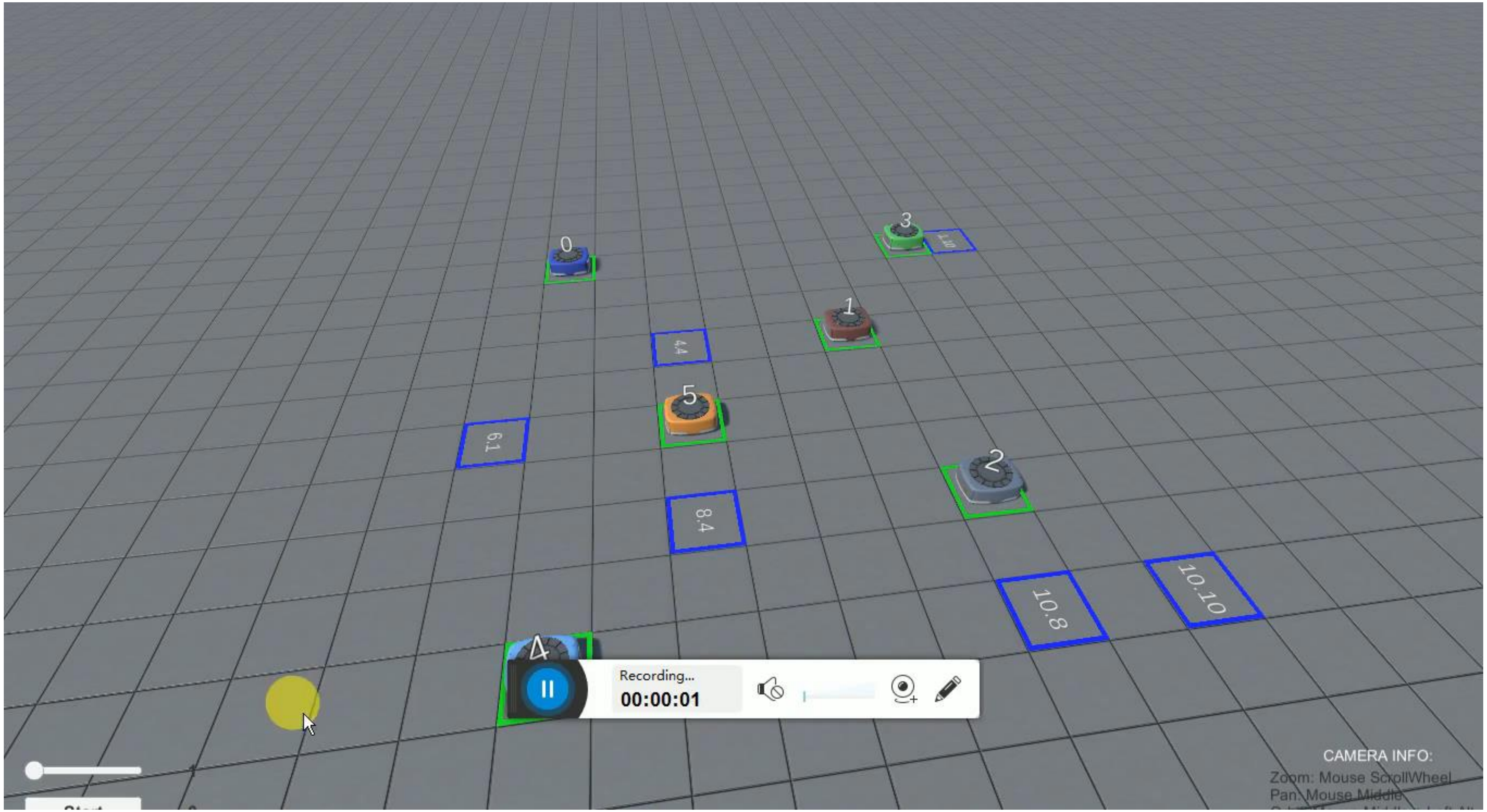


Results

Four images, of the same trained AI.



Results in Unity 3D



Conclusions

- An interesting example of problem where a simple neural network was coded has been shown

- The goal was reached only by a simple laptop and with a normal home pc

- New things can be added at the problem putting new goals at the problem

Thanks!

Thanks everyone!