Deep RL Arm Manipulator

A. Cámara

Abstract: In this project a DQN agent is created with its corresponding reward functions to teach a robotic arm to touch a goal object with its gripper. The system is tested with a simulated environment in Gazebo.

Introduction

Deep Q-Networks (DQN) were introduced by DeepMind in 2015 as a combination of Deep Neural Networks and Reinforcement Learning. They can directly deal with high dimensional states like images and learn from them like humans and animals do. This project uses a LSTM recurrent neural network to implement a DQN agent that learns to manipulate a robotic arm. The agent is defined in PyTorch working in a Nvidia Jetson TX2 with Gazebo. Appropriate reward functions are defined to make the learning possible pursuing two objectives touching the goal object and touching it only with the gripper.

Reward Functions

Reward functions are designed depending on the goal of the learning. They calculate positive or negative rewards depending on the results of the actions and its consequences toward the goal.

- Reward for touching the goal: For the first objective (touching the goal object) the reward function gives a reward for wining (defined with 200 points) and ends the episode when the object is touched. For the second objective touching with the gripper gives a win reward and touching with other part of the arm gives a loss reward (-200 points). In both cases the episode ends after touching the goal. These rewards allows the DQN agent to learn to touch the objective appropriately.
- Reward for hitting the ground: to avoid hitting the ground a loss reward ending the episode is given.
- Reward based on the distance to the object: this reward is calculated with a smooth moving average of the delta of the distance to the goal using a low alpha (0.1) to give more weight to the delta distance: average_delta = (average_delta * alpha) + (delta_dist * (1 – alpha)).

The joint control implemented is a position control. A function of the action (action / 2) has been used to find which joint's value to change what allows to have more degrees of freedom without changing it. The position is increased or decreased depending on the action value been even or odd.

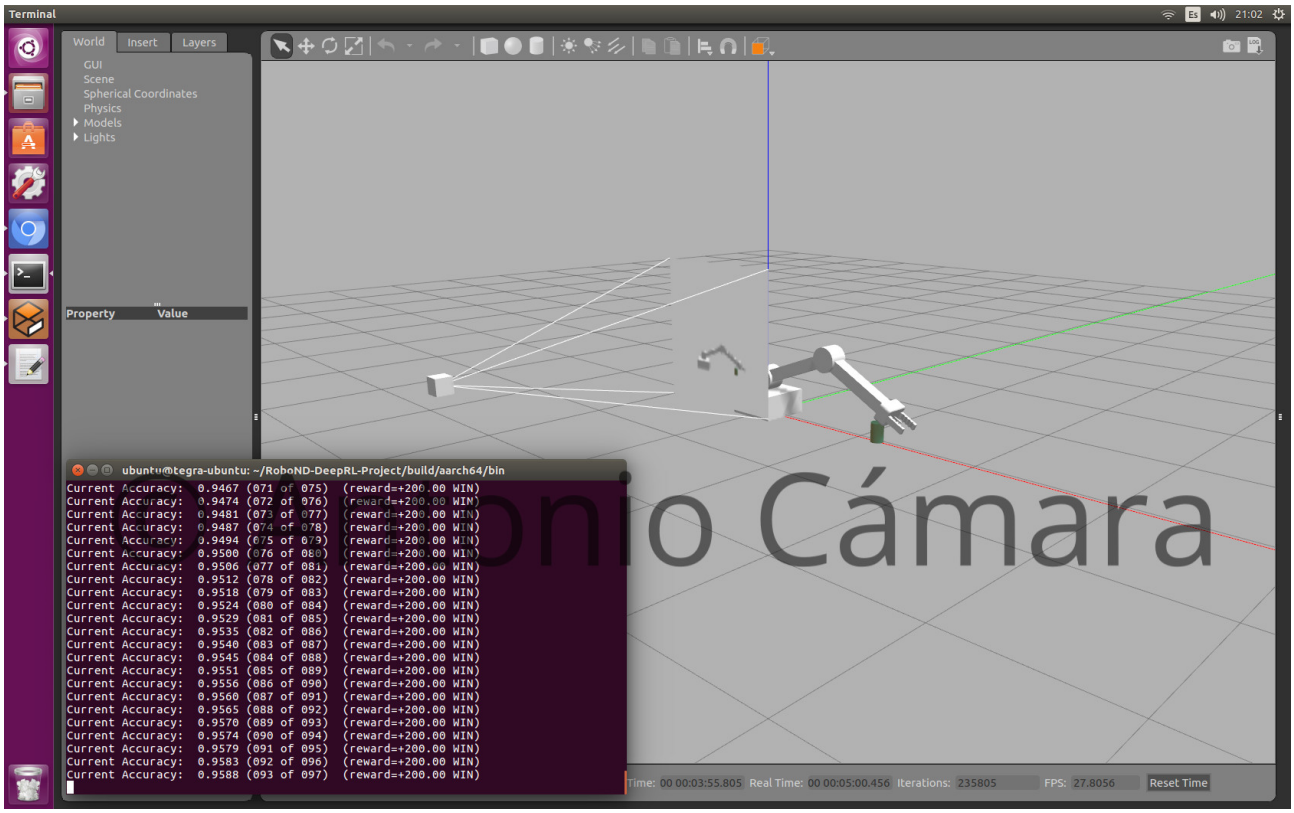
Hyperparameters

Several parameters have been defined to configure the algorithm to get good learning results. The same parameters are used for both objectives.

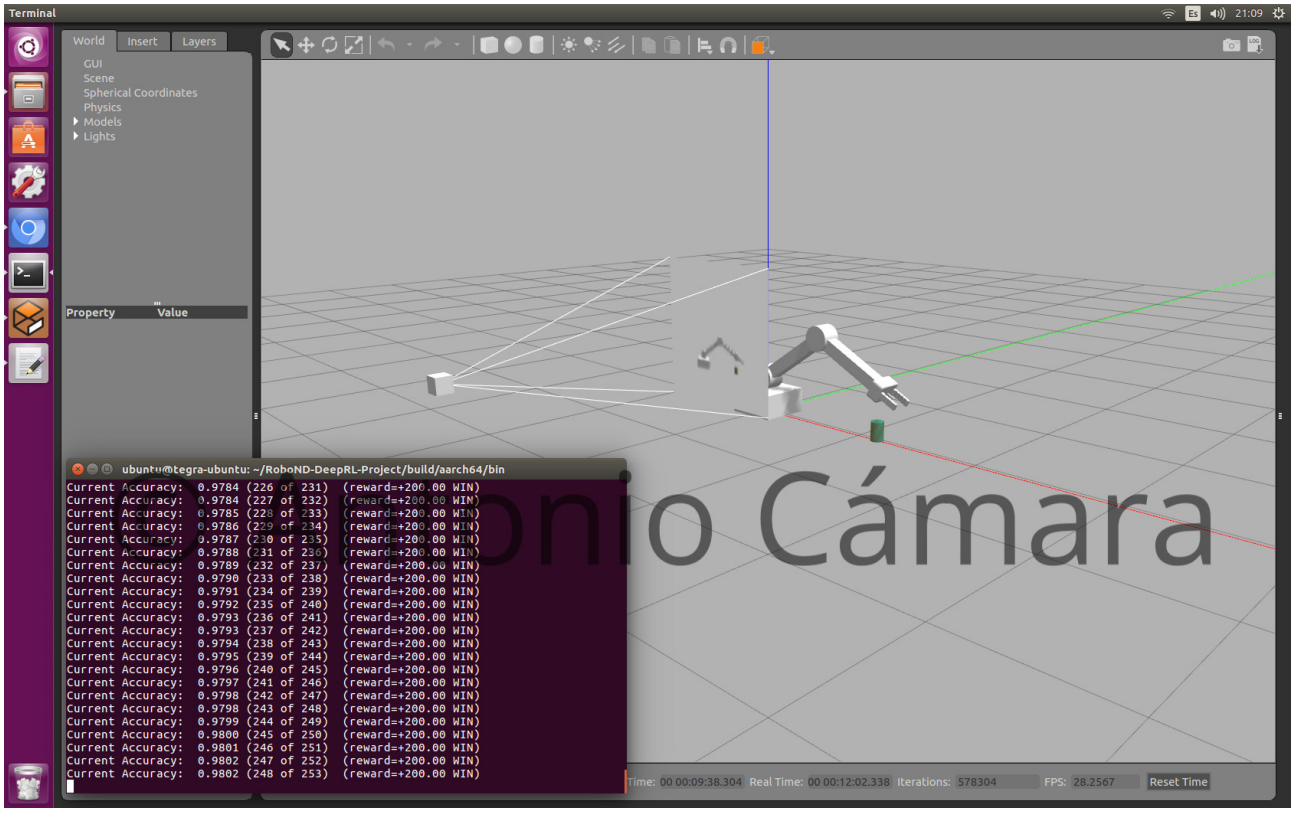
- Reward for wining and losing: big quantity of points are given for the events of wining and losing (200 and -200 points respectively).
- Input width and height: 64x64 images have been used. That is enough resolution to see the arm and goal and small enough to allow a fast training using the TX2 platform.
- LSTM have been used for the DQN agent with a size of 256, a batch size of 32 and a replay memory of 10000 what gives good results in other DQN learning projects implemented in the Jetson TX2.
- A high learning rate of 0.1 have been used to allow a fast convergence.
- RMSProp optimizer have been used as it is used in other DQN learning projects implemented in the Jetson TX2 which allow a good training optimizing only the learning rate.

Results

After configuring reward functions and learning parameters, results for objectives are checked. For the first objective (any part of the robot arm should touch the object with at least an accuracy of 90%) we get good results that accomplishes the objective very soon and get results of 98 % after 250 runs.

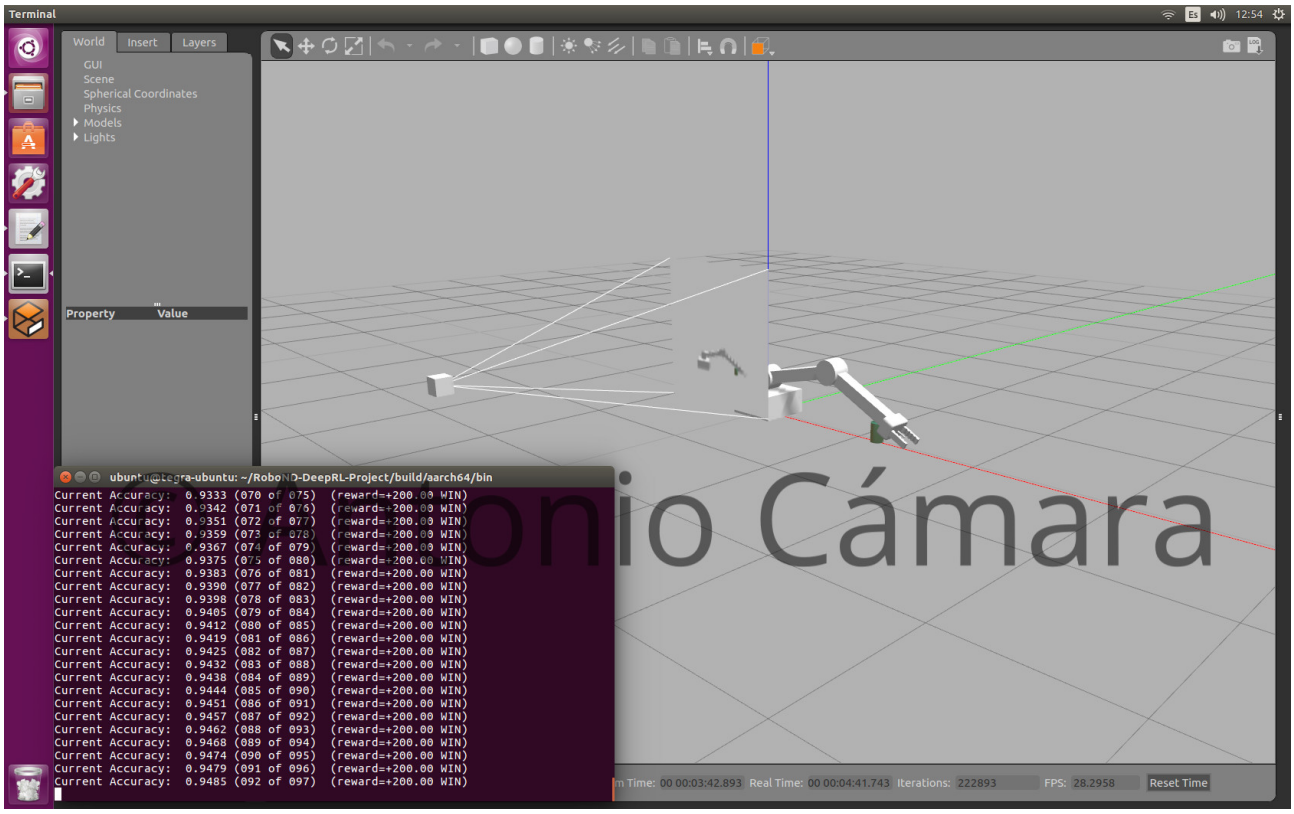


Near 96% in 100 runs

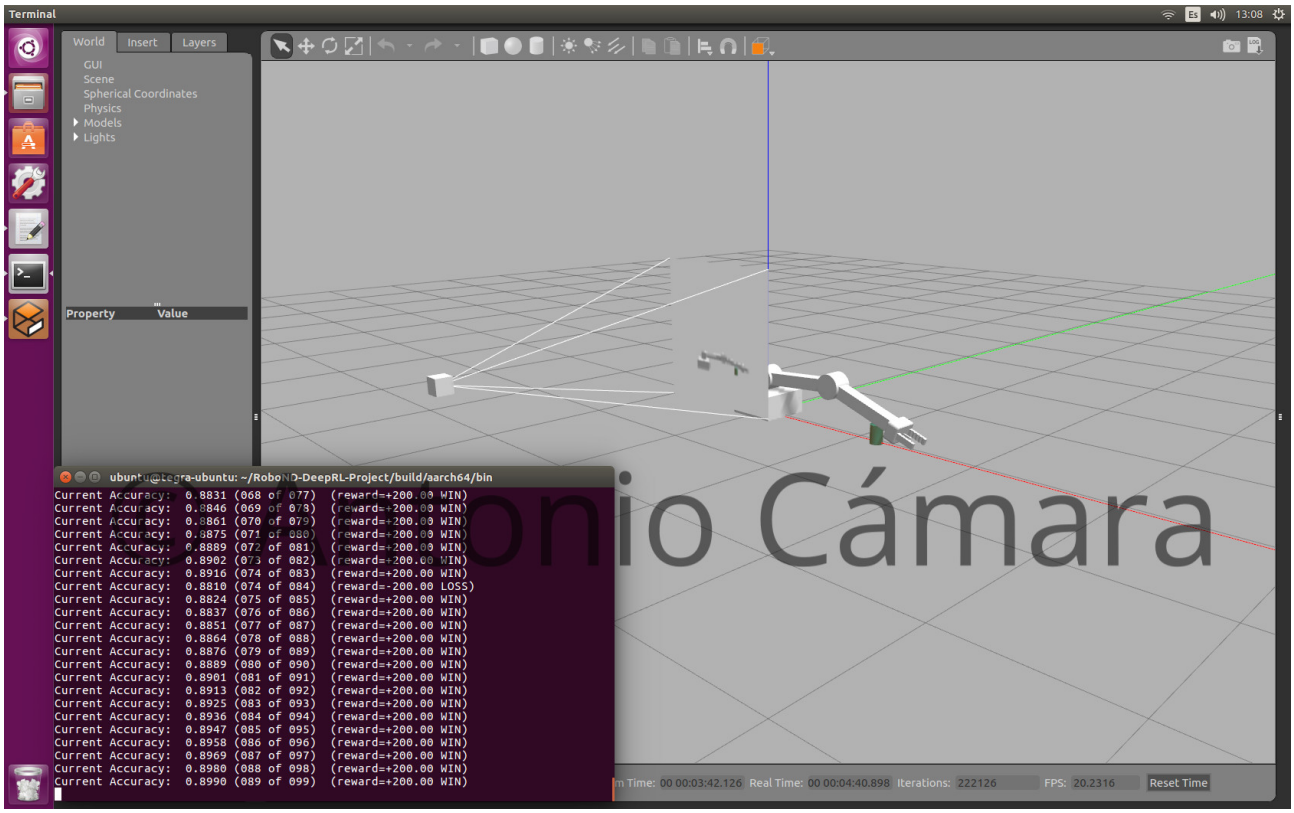


98% after 250 runs

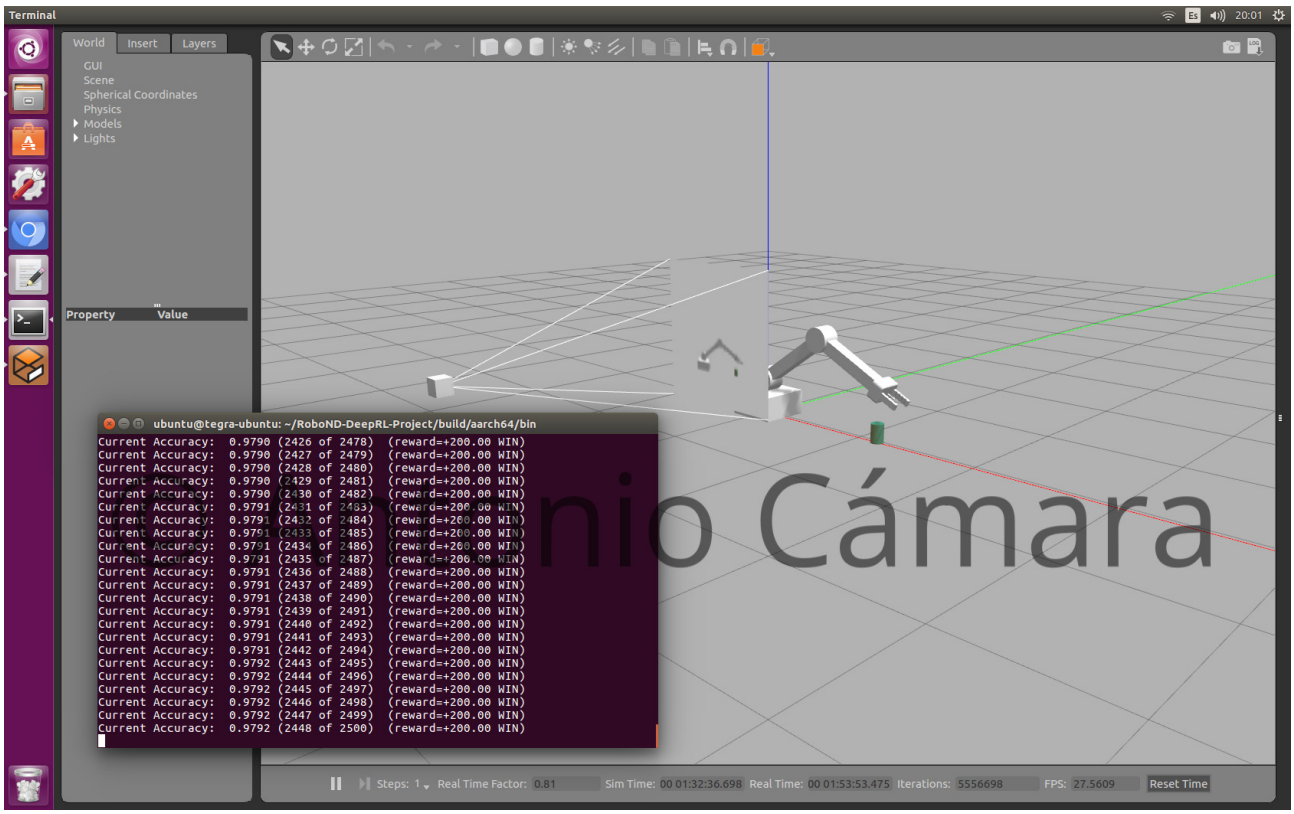
The second objective is fulfilled very well too but with worse results than the first. Below two training sessions are shown with 94% and 90% of accuracy after 100 runs. Another example is provided getting 97% after a long training session of 2500 runs.



94.8% after 97 runs



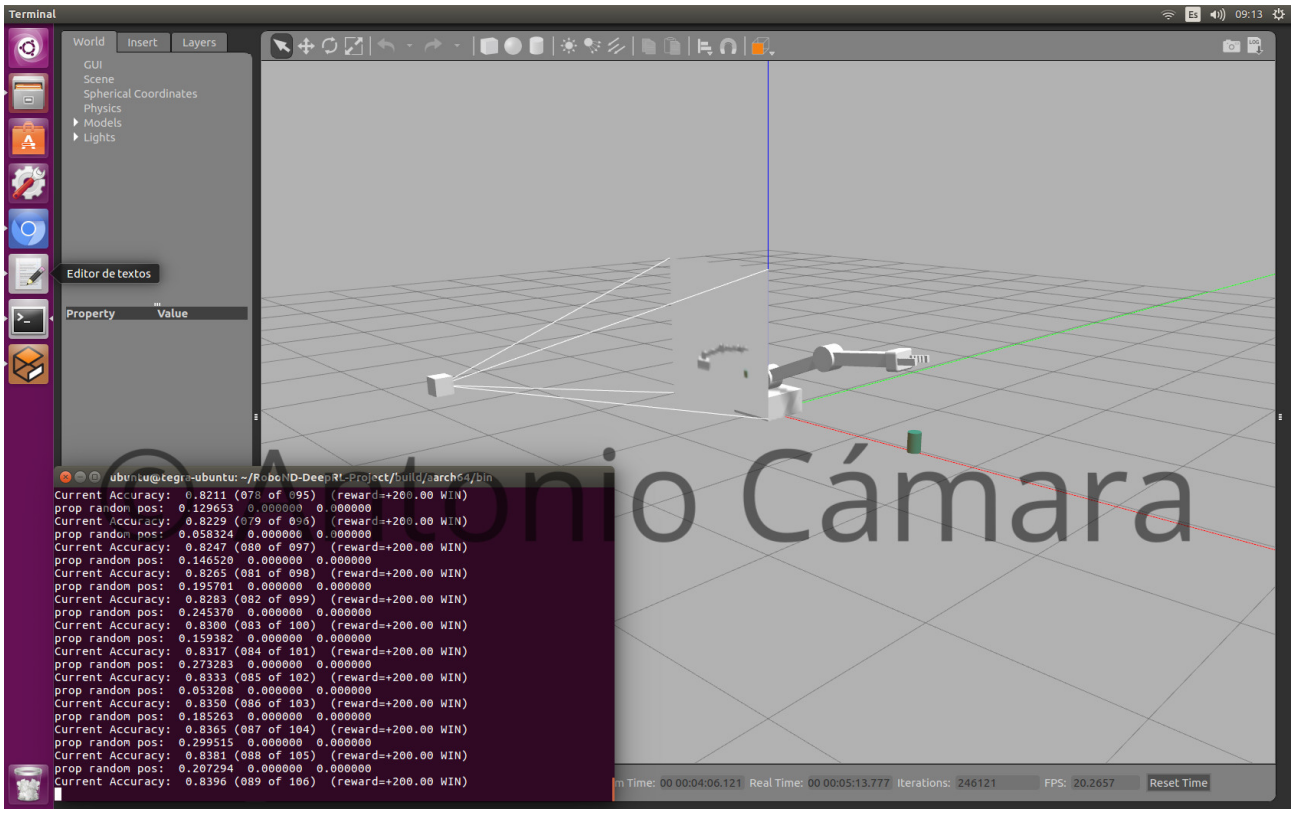
89.9% after 99 runs



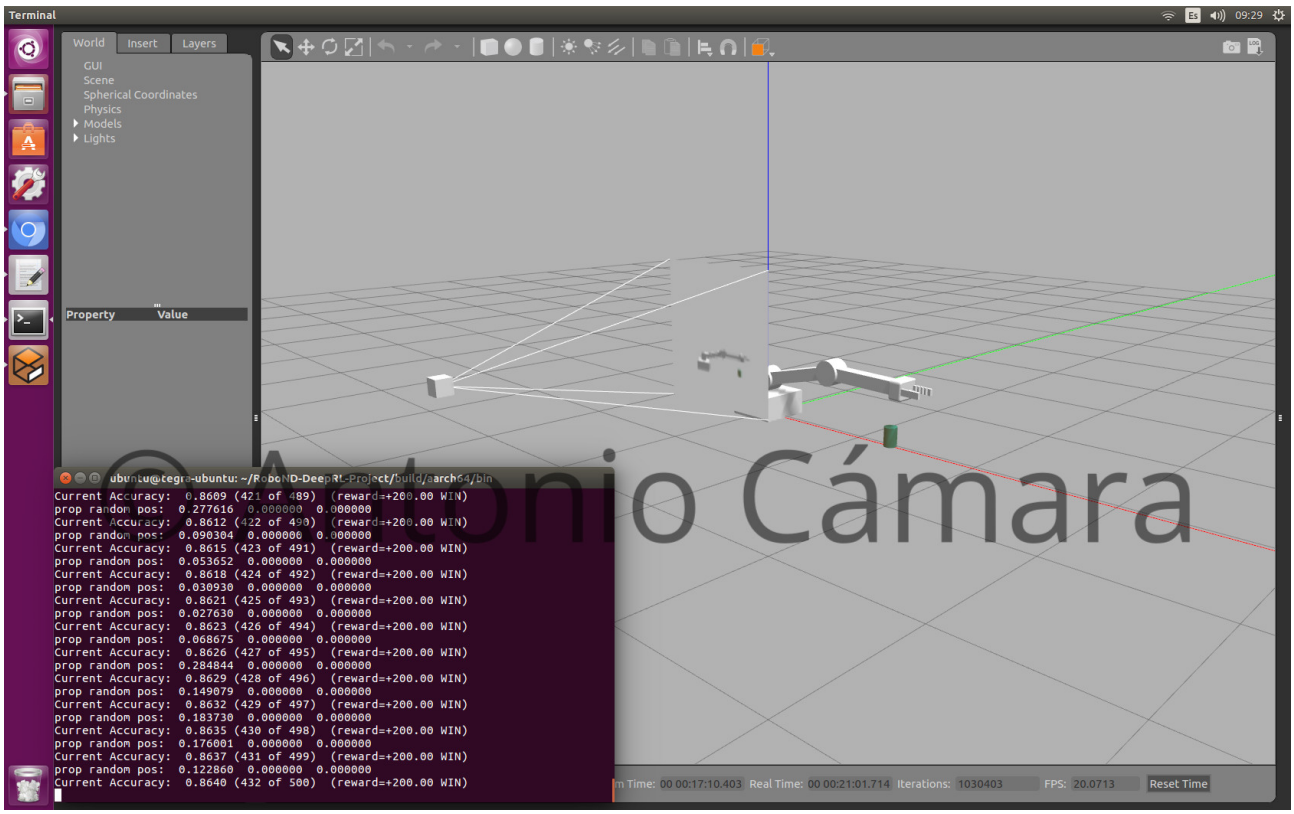
97.9 % after 2500 runs.

Additional Challenges

An x position randomization is added in first place. The object has different locations along the x-axis. This test gives an 83% accuracy after 100 runs increasing to 86% after 500 runs.

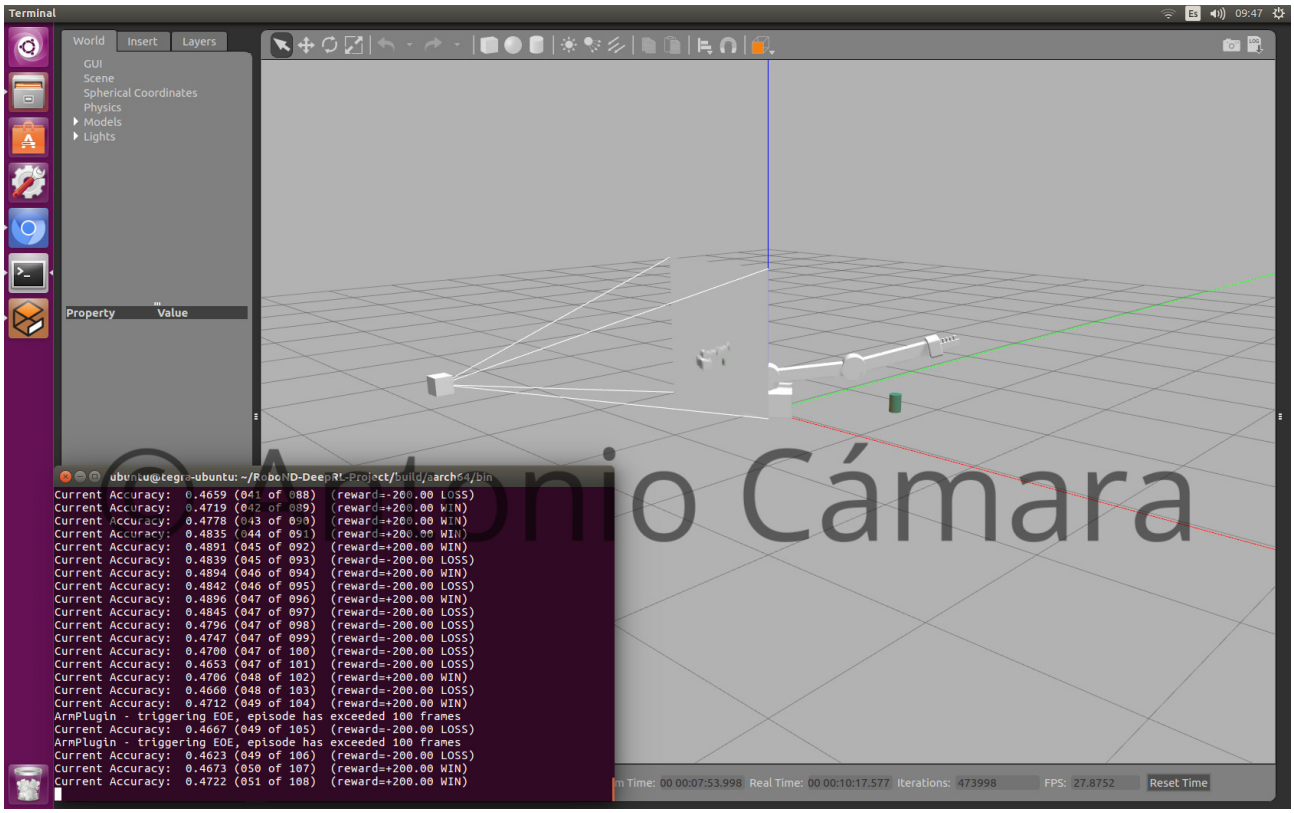


83% accuracy after 100 runs.

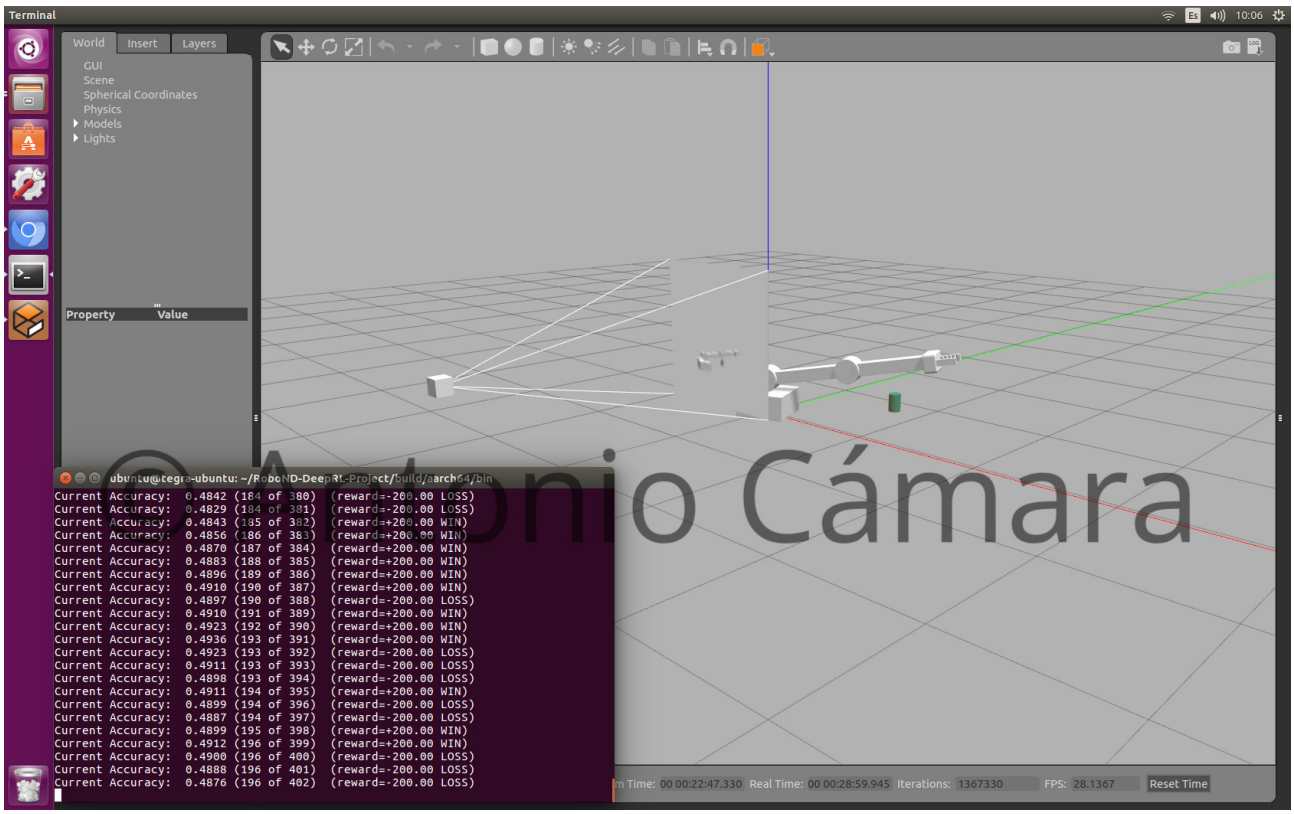


86% accuracy after 500 runs

In the second challenge the arm is able to rotate about its base and the object position is fixed in a position of (0.75, 0.75, 0). In this test a lower accuracy is obtained. 47% of accuracy after 100 runs and oscillating between 48% and 50% after several hundreds of runs.

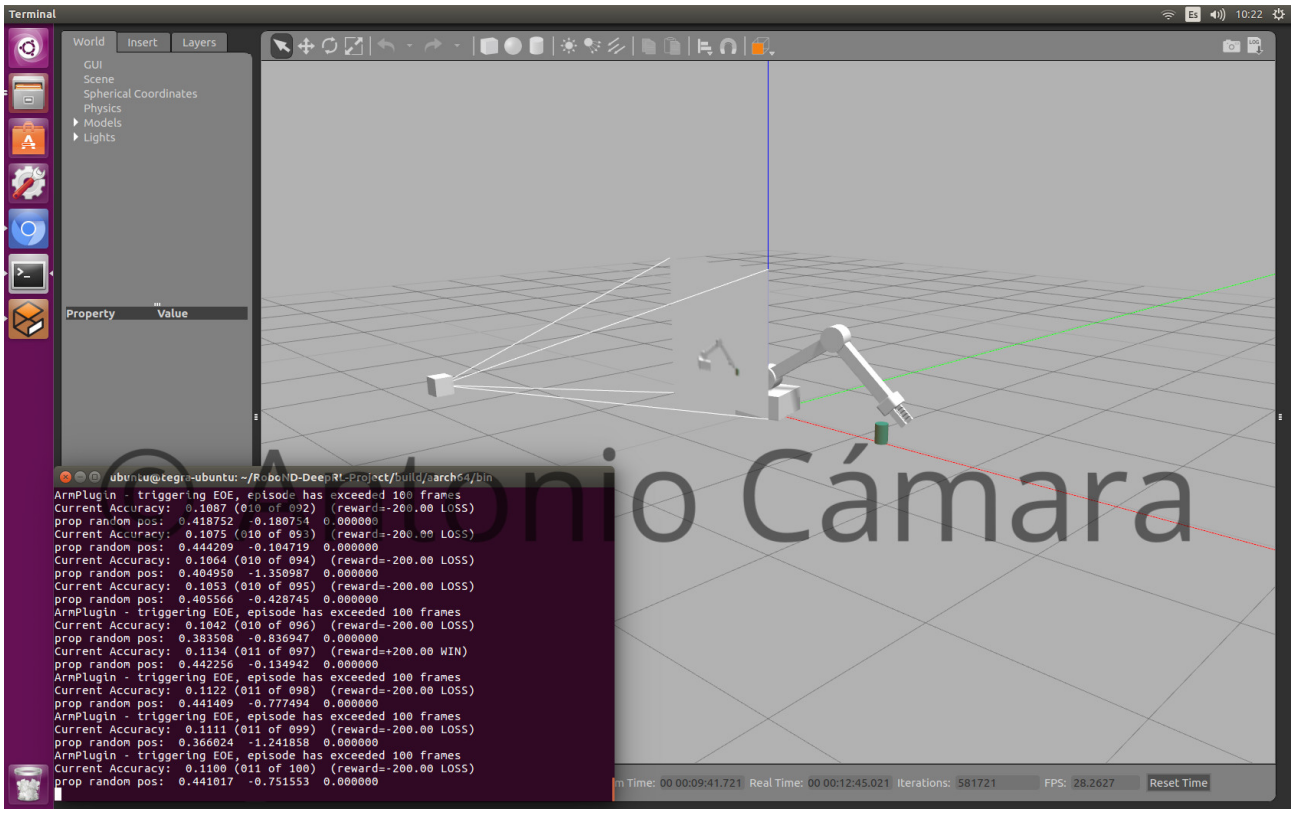


47% accuracy after 100 runs



49% accuracy after 400 runs

In the last challenge the object has a random x and y position with the arm been able to rotate. In this test the accuracy obtained is the lowest, 11% after 100 runs.



11% accuracy after 100 runs

Conclusion / Future work

A DQN agent has been trained to learn two objectives with a robotic arm with good results. In both objectives a 97% accuracy has been achieved after enough runs. With additional challenges the accuracy has been reduced due mainly to the lack of information about the goal position in captured images. This implementation can be adapted to other robotic control applications defining appropriate reward functions.

As a future work, a more complex control can be done. Challenges' results can be improved with another image taken from the top to see the x and y position of the goal object. Additionally, a complete pickup task can be done completing the reward functions needed in each state.
For a more precise training, a bigger resolution can be used having into account the longer time needed for training.