RUTGERS

Rutgers - Computer Graphics Course - Assignment C1

Unity ML-Agents Toolkit: Static Obstacles

This is an optional group assignment.

Total Assignment Points: **15**

INSTRUCTIONS

Based on your work from Assignment C0, evaluate the performance of your RL agent on a 50 meter by 50 meter plane with goals uniform-randomly sampled from the total area based on the Percentage-Above-Optimal (PAO) averaged over 50 instances of this environment.

$$PAO = \frac{d_{walked} - d_{shortest}}{d_{shortest}} \times 100 \qquad (3)$$

The PAO metric is a measure of inefficiency when navigating to a destination that is calculated by Equation 3, where $d_{shortest}$ is the shortest distance to the goal and d_{walked} is the actual distance that the agent travelled to the goal.

In our case, the shortest distance is the Euclidean distance between the agent's initial location and the goal's location. The distance walked can be computed by aggregating the change in position while the agent is navigating between its initial location and the goal. Please **report the average PAO** for your training environment from C0 and the PAO for the new training environment on 50 different instances. **Describe the change** in PAO (if any) and **give your reasoning** behind why this occurred.

Next, introduce a parameterizable number of randomly positioned static obstacles that are in the shape of 1 meter by 1 meter cubes. The goal is to have the agent to navigate between its initial location and the goal without hitting any obstacles. **Report the number** of static obstacles that you trained using.

Train your model until you are satisfied with the performance. Afterwards, evaluate the average PAO for 50 cases with static obstacles and **report this value**.

The naming convention of classes is to end the name in "_G##", where the number is your group number.

SUBMISSION

Since this is an extra-credit assignment, points will be awarded at the graders' discretion based on the quality of the agent's navigation.

Your final submission (**ONE** per group) will include the following:

- Source code (i.e., for your agent subclass and any dependencies)
- A prefab file, where you have placed in it the Brain GameObject you used for training. The Brain should not be a child of the prefab, the prefab should have the Brain's components.
- The saved model which can be found in (`ml-agent-0.6.0a/models/.../*.bytes`). You must submit the file with extension ".bytes" that you used in your Brain.
- A video demo showcasing the result of the trained model.
- A complete description of your new observations, actions, and rewards; also all of the bolded instructions above.