IKT442

Drone simulation

# Introduction

Simulation is a great tool.

As it allows for an affordable way to for example train a self-driving vehicle. It also allows for quicker training, as the environment can be reset in seconds. While a self-driving vehicle which is not in a simulated environment, might have to be moved back to say a road or such using an individual resetting for example a car.

It is especially useful for reinforcement learning, as a score can more easily be extracted from a simulated environment.

This report attempts to create an environment which can be used for quick development of reinforcement algorithms.

# Background

Simulation is a widely used technique used to check how something acts in the real-world without having to worry about damaging equipment.

# Simulation

Simulation is an important tool, as it allows us to attempt something in a virtual environment which means that nothing is damaged as if the user collides or creates an accident in the simulation, it can be reset to revert to the same exact state as it was before the accident occurred. An example of this is simulating vehicles such as airplanes, in order for the pilots to learn how to fly a plane without risking the plane or others’ lives.

# Reinforcement Learning

When attempting to create AI which can learn many difficult environments letting the AI-algorithm learn without the user having to show the algorithm how to do something would be preferable, as the creator of the algorithm would have to gather training data. This all means that the algorithm is essentially impossible to implement if there is not enough data, there is also the issue of having to train the algorithm for each environment. This is where reinforcement learning comes in, it does not learn from a human but instead learns by acting at first randomly and eventually in a way to increase rewards. The rewards can for simulated environments often be read of the screen, which allows for automatic training which does not need humans gathering data and reading said data. This is why reinforcement learning is a great algorithmic choice to make when choosing types of AI where a reward can be inferred. Examples of which are simulations, in which a vehicle is attempting to get from point A to point B.

# Unity

One of the most used game engines, is called Unity. Unity includes shapes, which can be used to create objects, physics, which allows for a quick and good simulation of real-world physics. And the ability to create code which can be easily connected to objects. There also exist many different attributes to these objects, which determine if other objects go straight though this object, how interacts with physics, and much more. This means that in order to create a simple simulation we only need a plane, which can be flat. An agent such as a drone to interact in the environment. A starting point as well as a goal.

Everything from the camera to the drone’s movement can be controlled by small scripts, which eases the implementation as each object can be linked easily. The linking of an object to a reference in the code, is done via GUI, by dragging the object to the reference name in the script’s attribution list.

Seeing how Unity is so widely used, many people will be easily able to change the project to suit their needs. It may also make it simpler to further develop the simulation.

## Terrain

Terrain is an important part of the simulation, as more complex terrain will lead to the environment being more difficult to traverse. This leads to a question of how to generate these terrains, as it is time-consuming to create many different terrains in order to create different difficulties of terrain for the agent to interact with. This means that it is better to be able to create terrains on demand.

Luckily Unity allows for the modification of a terrain through code. Meaning that the creation can be done automatically. Which entails for the difficulty of the environment to be changed as the agent gets better at interacting with said environment.

This can be done using a script with just a few variables. The environment needs to be a certain width and breadth. The depth or the maximum height which creates the height differential between points. The scale which tells the engine how quick changes are, or how random the environment is. Followed by random variables which using math is smoothed to create the random function which allows for different scenarios for every run.

## Drone

The drone or the agents, is used to interact with the environments. In this report it will be based on a common quadcopter. Meaning that there are three wings with engines connected to them.

There are a few things which need to be scripted for the drone, the movement and the camera which should follow the drone.

The movement can be programmed in a couple of different ways, one of them is merely adding a force to the drone object for its movement. This how ever means that the drone does not rotate in the same way as a real drone would. Another way is to add the force not to the drone object, but instead the rotators connected to said drone. This allows for a realistic simulation of the drone, which can be fine tuned using different forces for different weights of the drone.

The camera also needs to be programmed as it is important that the agent is able to see what its movements does. This is done by simply saying where the camera should be as an offset of the drone. Meaning the camera could always be in the front of the drone for example, or say giving the agent a third person perspective, by placing the camera above and behind the drone.

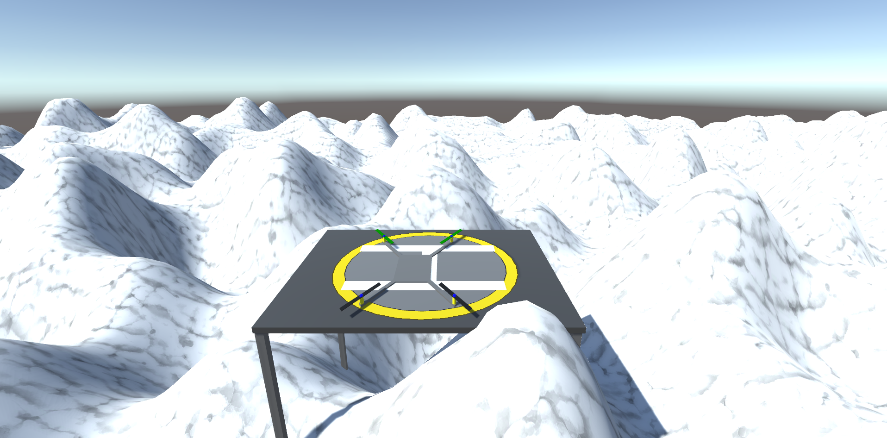
## Reward

If the simulation is going to be used for reinforcement learning, then there needs to be a reward giving the reinforcement algorithm feedback on its actions. This can be done easily in unity, by calculating the distance between the drone and the goal. Then showing a positive number if the drone is going towards the goal, or a negative number else. If the distance is the same the drone would then not be given either a reward or penalty through the environment. Though this could be done in the reinforcement algorithm itself.

When the reward is displayed the agent is then able to simply use OCR or Optical Character Recognition to read said reward.

The goal is created more easily, as it needs to be a place for the agent to go, resulting in a large reward. It does not need to be dynamically generated, though it should be dynamically placed, given that the environment is changing.

Pic 1: Drone in flight



Pic 2: Drone resting