# COMS4047A Assignment
# Obstacle Tower

Your assignment for this course is to create an agent to tackle the *Obstacle Tower Challenge* (`github.com/Unity-Technologies/obstacle-tower-env`). Obstacle Tower is a procedurally-generated game in which an agent must ascend a tower, proceeding through as many floors/levels as possible. On the first few floors, the agent must simply find and pass through the door that leads to the next floor. On later levels, the agent must first find and collect a key, which can then be used to open the door leading to the next floor.

This is a group assignment – **groups of 3 or 4 are allowed**. You may implement any method you choose, and use any kind of pretraining or extra information to train your agent. You will then submit your trained agent on Moodle, and it will be evaluated on a number of unseen levels.

Along with your code, you should submit a video that explains your method and shows your agent in action. The video should be similar to research videos that are often produced alongside publications. The video should clearly explain and demonstrate your approach (including any tweaks or modifications you implemented), and show how they improve your agent). Examples of videos can be found here (`https://www.youtube.com/watch?v=ChxSx8-sX_c`) and here (`https://www.youtube.com/watch?v=4Qg5I5vhX7Q`):

This is a competitive assignment, and so your aim should be to maximise your agents performance the better it performs, the better your mark. There also be a reference agent that implements Deep Q-Learning with a large exploration term, which you can use to judge the performance of your agent. You should expect to outperform this reference agent. The final mark is a combination of your agent's performance, and the hand-in video.

More information about the assignment will be communicated to you in the near future. To get started, please see the instructions below.

# 1 Installation

First setup a python environment with the following requirements:

- **Python 3.6**
- Unity ML-Agents v0.82 (`https://pypi.org/project/mlagents/0.8.2/`)
- OpenAI Gym
- Pillow

Then, download and unzip the game binary for your operating system:

- Windows: `https://storage.googleapis.com/obstacle-tower-build/v2.2/obstacletower_v2.2_windows.zip`
- Mac OS: `https://storage.googleapis.com/obstacle-tower-build/v2.2/obstacletower_v2.2_osx.zip`
- Linux: `https://storage.googleapis.com/obstacle-tower-build/v2.2/obstacletower_v2.2_linux.zip`

**Note that for this assignment we will be using version 2.2, due to the bug in the Linux version 3.0**

Finally, download the environment and random agent from Moodle, and make sure you are able to run the code.

# 2 Environment Specs

We will be using Unity's Obstacle Tower wrapped by the OpenAI gym interface in *retro* mode. To create the environment, use:

```
env = ObstacleTowerEnv('path_to_binary', retro=True, config=config)
```

The environment can be configured in a variety of ways—both when launching the environment, and on episode reset.

```
env = ObstacleTowerEnv('path_to_binary', retro=True, config=config)
```

or

```
env.reset(config=config)
```

We will use the following settings for this assignment:

- starting-floor: 0
- total-floors: 9
- dense-reward: 1
- lighting-type: 0
- visual-theme: 0
- agent-perspective:1
- allowed-rooms: 0
- allowed-modules: 0
- allowed-floors: 0
- default-theme: 0

## 3 Implementation

You can train you agent using any method you would like to. Your fully-trained agent should be submitted to Moodle where it will be evaluated. Note that no further training is permitted once your agent has been submitted.

In order to evaluate your agent, we will create an instance of the Agent class. You need to implement your agent within this class. Ensure that all your pre-processing is done on initialisation method. This includes loading any PyTorch/Tensorflow models inside the `__init__` method.

The one other method that should be implemented is the act(obs) method. This method accepts an observation from the environment, and returns an action. Your agent should decide on the best action to take and return it.

## 4 Evaluation

We will evaluate your agent on a number of runs and average the total reward. The levels your agent will be tested on are unknown, and so you should ensure your agent is able to generalise to unseen levels (and not overfit to those you train on). The score will be based on the number of levels cleared. If two agents have cleared the same number, then the reward obtained will be used as a tie-breaker (agents receive reward for collecting important objects and for clearing levels).

## 5 Submission

Zip your code along with all required files (such as model file, that needs to be loaded on initialisation of agent) and upload it onto Moodle for marking. You should also upload your video under the relevant link.