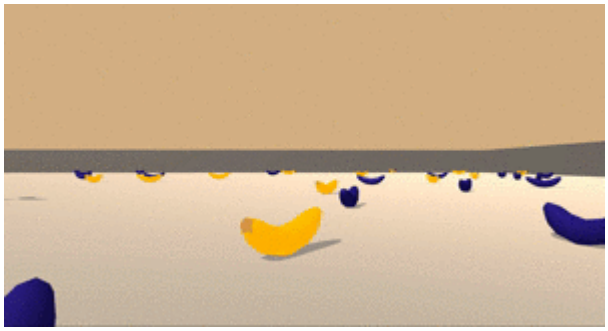
Project 1: Navigation

Introduction

For this project, we will train an agent to navigate (and collect bananas!) in a large, square world.



A reward of +1 is provided for collecting a yellow banana, and a reward of -1 is provided for collecting a blue banana. Thus, the goal of the agent is to collect as many yellow bananas as possible while avoiding blue bananas.

The state space has 37 dimensions and contains the agent's velocity, along with ray-based perception of objects around agent's forward direction. Given this information, the agent has to learn how to best select actions. Four discrete actions are available, corresponding to:

- `0` - move forward.
- `1` - move backward.
- `2` - turn left.
- `3` - turn right.

The task is episodic, and in order to solve the environment, the agent must get an average score of +13 over 100 consecutive episodes.

Setting Up the Environment

There are three environments we are talking about here:
1. Operating Environment - This is the hardware / system software used to design/run the program.
2. The Python Environment - This is the programming environment with necessary packages to train the agent.
3. The Unity Environment - This is the game environment representing the world for our agent.

Operating Environment

This script has been developed on a laptop with the following main specifications:

1. Processor: Intel Core i7-10750H CPU @ 2.6GHZ / 16GB RAM
2. GPU: NVIDIA GeForce GTX 1650 Ti / 4GB
3. Operating System: Windows 11 Home Insider Preview, Version: 24H2

Python Environment

- This script has been developed on python 3.6. Updated versions of python couldn't be used, because the provided *Unity Environment* runs on package versions that are supported max on Python 3.6
- We hope that at some point in time, *Udacity* would provide an updated *Unity Environment* to allow for running on more up to date python versions.
- At the time of developing this script (April-2025), python 3.6 has already achieved end of life.

Given the above, the relevant python environment files are provided in python_env that should be used as follows:

1. Open Command Prompt in admin mode to install ' by pasting below lines one-by-one

```
C:\> Set-ExecutionPolicy Bypass -Scope Process -Force; `
      [System.Net.ServicePointManager]::SecurityProtocol = `
      [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; `
      iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))
```

2. Reopen Command Prompt to check that 'chocolately' has been installed:

```
c:\> choco --version
```

3. Reopen Command Prompt in admin mode to install 'SWIG':

```
c:\> choco install swig -y
```

4. Reopen Command Prompt to check that 'SWIG' has been installed:

```
c:\> swig -version
```

5. Open Anaconda Prompt and create conda environment using the file DRL.yml:

```
(base) c:\> cd <path_to_python_env>
(base) c:\path_to_python_env> conda create --name DRL python=3.6
(base) c:\path_to_python_env> conda activate DRL
(DRL) c:\path_to_python_env> conda env update --file DRL.yml
```

6. There are other packages that also need installation, but using *pip* and not *conda*. While we could have added those packages to our DRL.yml, but the issue is that we want to install those *pip* packages in sequence. *conda* environment files collects all *pip* dependencies together and then installs them together, which fails with those packages. This is why we created a *powershell* script DRL.psl to install those packages in sequence.

```
(DRL) c:\path_to_python_env> powershell -ExecutionPolicy Bypass -File DRL.ps1
```

7. For completion purpose we provided the file DRL.bat which could be used to both update the environment (after creating and switching to it) and install the remaining *pip* packages. However, we found that after executing the first command in the file, the script stops. As such the script is just provided as a reminder of the commands to use.

Unity Environment

1. Download the environment from one of the links below. Only select the environment that matches the operating system (for our installation, we needed the Windows(64-bit)):
    - Linux: click here
    - Mac OSX: click here
    - Windows (32-bit): click here
    - Windows (64-bit): click here

2. Unzip (or decompress) the file and place the folder Banana_Windows_x86_64 under the root of the project.

Instructions

- As we are more oriented toward python scripts more than jupyter notebooks, we are using the Navigation.py as our driver script.
- While we are using *PyCharm - Community Edition* as our IDE, the script can be run from a python terminal using
  ```
  python navigation.py
  ```
- To know more about what the above command will do (how the project is structured), please refer to the attached report in either md or pdf