Code Tutorial
Multi-Agent Modelling

Embodied AI, Vrije Universiteit 06/2020

General Information

All the simulations will be run in python by utilizing a python package known under the name **Pygame**. In brief, Pygame is a set of python modules originally designed for writing video games (see link: Pygame Official Documentation). In this course we are going to use it to simulate multiple agent interactions. The python version used in the course is **python 3.x**. We recommend using python version **python 3.6**, as the code has been tested with this version, but other python 3 versions should also be compatible.

Pygame Installation

For correct installation please follow the official installation guide by Pygame, link: Pygame Installation. There it will be specified the correct approach for each operating system type: Mac, Windows, and Linux.

Other Packages

Apart from PyGame you should also have installed numpy (Numpy Install) library.

Folder Set-Up

The overall structure of the code is depicted in figure 1. Under the main project folder, *EmbodiedAI*, you will find the main execution file, *main.py*, as well as 2 sub-folders, *simulation* and *experiments*.

- main.py : The main execution file. Here you initialize pygame, the simulation with its corresponding settings from the parameters.py file, as well as set the simulation to 'running' mode.
- simulation folder : The folder contains general properties of a swarm or an agent that can be re-used across different types of multi-agent modeling experiments. You should not alter the code within this folder, but you can use its classes and methods for your own experiments.

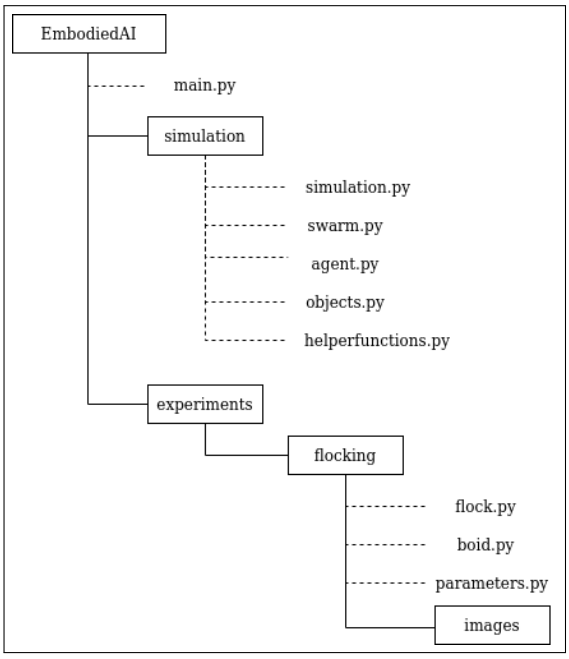

Figure 1: General structure of the code

- experiments folder : At the moment the folder contains only one sub-folder, Flocking. The flocking code is given to you as an example template of how to use the existing classes within your own implementation, thus throughout this course you will implement your own experiments and store them here under the experiment folder with a new sub-folder name.

Simulation Folder

In this section you will find a brief description of the python files you can find in this folder. We recommend reading this section while looking at the code for a proper understanding, because we will reference some of the functions that are presented in the code. Once more, the content of this folder should not be altered, this explanation below is given to you so you can understand what utilities are already implemented for you.

simulation.py

This python file contains a class called **Simulation**, which is the general simulation loop implementation. In general, you should not change anything within this class.

swarm.py

In this python file you will find a class **Swarm**. This class should be used as the underlying super-class for any specific swarm implementations. In the code you can see the variables this class can hold (agents and objects), as well as that its initialization requires only one parameter (screen size). The remaining function names are quite self-explanatory. Just one minor clarification, you use the 'update general' function to update the agents, and the 'display' function to actually display the changes made.

agent.py

In this python file there is a class **Agent**. Similarly to the previous explanation, this class should be used as the super-class for any other agent type implementations. This class contains a lot of parameters that can be passed to it during initialization allowing it to be versatile, thus, it can be adjusted depending on the needs of a given agent type. Functions that might be useful for your own implementations are:

- wander : creates a simple 'random-walk' like behavior
- avoid-obstacle : obstacle avoidance (but very limited)

objects.py

Contains a class **Objects**. The class can hold 2 types of objects, obstacles and sites. This class is useful when you want to add your own custom objects to the environment. In particular, you can add objects via calling the method 'add object', where you must specify the type of object to

add (obstacles or site), its position in the environment (middle coordinate points), file (image to load), as well as whether you want to apply scaling factor if the image is very small or too large. Important to point out, that the image you load MUST have a transparent background.

helperfunctions.py

Contains some useful methods for vector transformations/normalizations/computations.

Flocking Folder

As the name of the folder implies the folder contains a swarm and an agent definition specifically related to the flocking behavior observed in birds. By carefully examining the structure of this folder you should be able to create your own experiments in a similar fashion. Each experiment folder should contain 3 files: a specific swarm type (in this case flock), a specific agent type (in this case boid), as well as best parameter settings for the simulation (parameters.py file). In the sections below you can find a more detailed description of the code structure for the flocking experiment.

flock.py

In this python file you will find a class called **Flock**. The Flock class inherits all methods that the Swarm class has. Hence, it builds upon the general methods with definitions specific for a flocking behavior. In particular, you can find here a method 'initialize', which defines where to position the boids within the environment and whether to create any obstacles in the environment as well. Furthermore, there are 3 other functions that are necessary to compute the neighbor boid forces (for further explanation see Assignment 1).

boid.py

This file contains a class **Boid**, which is a subclass of the Agent class. Important function to pay attention to is the 'update actions' method: this function defines on how to 'update' the agent velocity by computing a steering force. In general, any new agent definitions that you will create should include this function. In your own implementations you can change its internal structure, but the name of it should be left unchanged. The remaining functions in this class are used to compute the 3 forces needed to model flocking: alignment, separation and cohesion.

parameters.py

The parameters file specifies (i) the general settings of the simulation, (ii) the specific flock settings (the environment where they act), and (iii) the specific boid settings, such as, the flocking behavior forces.