# Assignment: Behaviour Trees

## IMPORTANT: For this assignment, be sure you are using version 0.1.1-beta of the AAPE environment. You will find it in the virtual campus in "Practice environment", "0.1.1-beta-AAPE environment"

In this assignment, we will use Behaviour Trees to implement a more complex behaviour in the AAPE platform. As explained in class, we will use our CritterMantaRay (`"type": "AAgentCritterMantaRay")` agent and the py-Trees library. Additionally, we will employ another agent, the astronaut (`"type": "AAgentAstronaut")`, to evaluate the critter's behaviour.

We want the critter to show the following behaviour:

- By default, it roams around, just avoiding obstacles.
- The critter has a flag that indicates if it is hungry or not. This flag turns off after staying near a flower for 5 sec. and goes on again after 15 sec. (be aware that this behaviour has to be programmed by you).
- While roaming around, if it detects a flower and it is hungry (hungry flag True), it goes straight to the flower and stays there for 5 sec. After that, we consider the agent satiated, and the hungry flag goes off. Then it starts roaming again. The hungry flag will activate after 15 sec (feel free to modify the length of these behaviours if you consider it necessary).
- If, at some moment, it detects the astronaut, it starts following it until it either is hungry and finds a flower (at that point, it will start the behaviour associated with the flowers described before) or it loses contact with the astronaut and starts roaming around.
- BONUS 1: Introduce a specialised search behaviour that makes the critter search for the astronaut when it loses contact. It continues the search for a certain amount of time and then gives up.
- BONUS 2: Use more than one critter in the scenario and give them an avoidance behaviour to be applied when they meet.

Figure-1 depicts the scenario "3-BehaviourTrees" that we will use for this assignment. You can still use the "1-Empty" and the "2-Avoid" scenarios if you need them to test specific goals or parts of the behaviour tree. However, the assignment's evaluation will be performed in the "3-BehaviourTrees" scenario.

As we said, to test the critter's behaviour, you can use the agent astronaut, which will be controlled manually using the keyboard.

As shown in Figure 1 (*), you can activate the manual control of an agent using the user interface and use the keys "WASD" to move the agent. This manual control can be used in any agent (be a critter or an astronaut) and can be activated/deactivated at your convenience. Notice that there is also the option to activate/deactivate the debug mode. When this mode is On/Off, it shows/hides the ray cast sensor rays.

Figure 1 also illustrates the spawn point locations, along with the corresponding numbers to be used in the AAgentX.json files (in the “1-Empty” and “2-Avoid” scenarios, there is only the “0” spawn point).

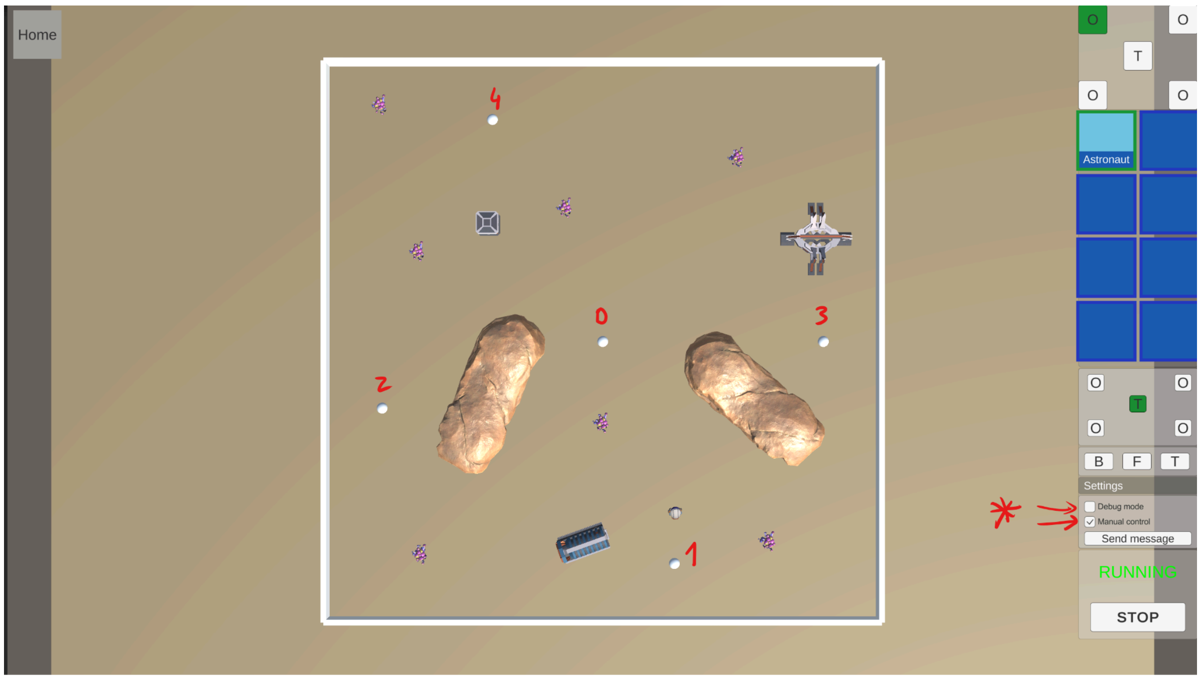
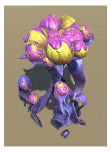


*Figure 1*

The tags used by the sensor for the different actors are the following:

 Flower Astronaut CritterMantaRay

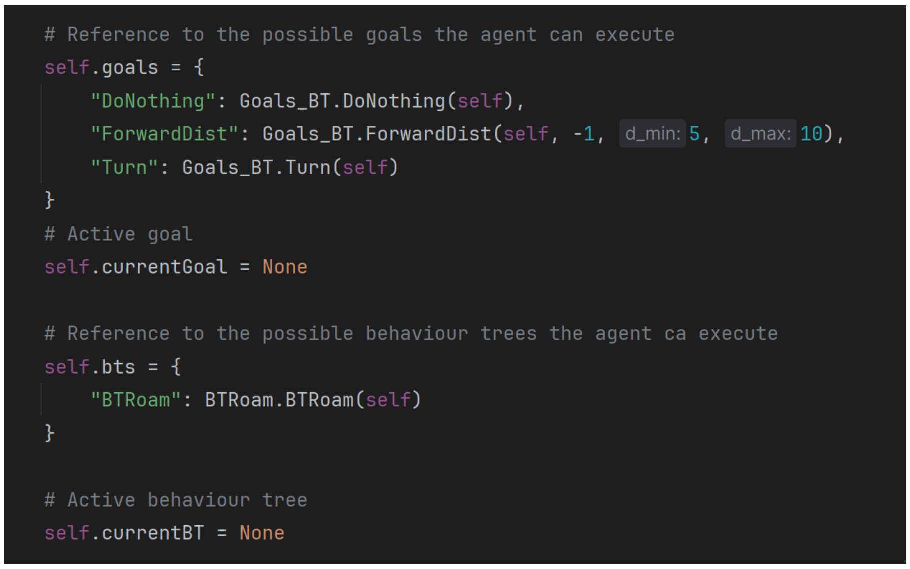
The rest of the elements of the scenario are tagged as “Rock” (each one of the two rock formations), “Wall” (the walls that delimit the perimeter) and “Machine” (the three sci-fi machines scattered in the scenario).

## Provided code

The file Goals_BT.py includes three example goals (DoNothing, ForwardDist, and Turn) that you can use as a reference for implementing goals suitable for use in a behaviour tree. In their current form, they are insufficient to meet the requirements of the requested behaviour. It is part of your assignment to adapt and extend those goals and implement other necessary goals. Additionally, an example behaviour tree, BTRoam.py, is provided. This is the one we discussed in class. Again, this is only an example to be used as a reference.

The AAgent-1.json is a configuration file for the critter agent, and the AAgent-2.json is the configuration file for the astronaut agent. Feel free to modify them at your convenience.

The AAgent_BT.py is the agent's main code. You only need to modify that file to register your goals and behaviour trees.



```python
# Reference to the possible goals the agent can execute
self.goals = {
    "DoNothing": Goals_BT.DoNothing(self),
    "ForwardDist": Goals_BT.ForwardDist(self, -1, d_min: 5, d_max: 10),
    "Turn": Goals_BT.Turn(self)
}
# Active goal
self.currentGoal = None

# Reference to the possible behaviour trees the agent ca execute
self.bts = {
    "BTRoam": BTRoam.BTRoam(self)
}

# Active behaviour tree
self.currentBT = None
```

*AAgent_BT code section where you have to register the goals and behaviour trees so they can be called from the AAPE interface ("Send message" button).*

## Submission Requirements:

Please submit the **AAgent_Python** folder compressed in a single zip file.

Apart from the files **AAgent_BT.py** (the version with your goals and behaviour tree registered) and **Sensors.py** (this must be the original file already provided). It should include:

**Goals_BT.py** file:

This file should contain your code for the goals used in your behaviour tree.

**BTCritter.py** file:

The code for the behaviour tree that implements the requested behaviour.

**AAgent-1.json** file:

Agent configuration file for the critter agent with your modification if there are.

**AAgent-2.json** file:

Agent configuration file for the astronaut agent (original file already provided)

**AAgent-n.json** files:

Any other agent configuration file you consider is necessary to test your implementation.

Please, use those specific names for the files.

**Documentation:**

Include a pdf document where you explain how you implemented the behaviour tree and highlight the main features of your implementation (5 pages maximum).

## Evaluation Criteria:

In addition to assessing the correct functioning of the behaviour tree as described, the following aspects will be considered:

**Originality:**

We will evaluate the uniqueness and creativity of your proposals for implementing the behaviour tree and the corresponding goals.

**Implementation:**

We will examine how effectively you have translated your ideas into code, considering factors such as efficiency, readability, and adherence to best practices.