Tips for the Behavior Trees Assignment – part 2

- Here you will find some tips for Implementing the goals of the Behavior Tree in the Alone Scenario.
- These suggestions are intended to help you and give you some ideas. However, they are not mandatory—you may choose alternative approaches if you prefer.

- For this scenario you have to be able to detect:
  - When the agent detects a flower

    Helping code:

```
# Iterate each ray of the sensor and check if a flower is detected
for index,ray in enumerate(zip(*self.rc_sensor.sensor_rays)):
     if ray[Sensors.RayCastSensor.HIT] and
        ray[Sensors.RayCastSensor.OBJECT_INFO]['tag'] == 'AlienFlower':
         print(f“Flower detected in ray: {index}”)
         print(ray)
```

  - When the agent detects an obstacle

    Use the same code before but compare the ‘tag’ to ‘Rock’ or ‘Wall’.

  - When the agent inventory is full

    Helping code:

```
# Get the amount of elements of type ‘AlienFlower’ from the agent inventory
alien_flower_amount = next(
    (item['amount'] for item in self.i_state.myInventoryList if item['name'] == 'AlienFlower'),
    0  # If not found
)
print(alien_flower_amount)
```

- Avoiding obstacles
  - Don’t overcomplicate that part—at least not in your first attempt. A simple approach is to use the central ray as a reference and turn to one side or the other until the obstacle is no longer detected. To decide which direction to turn, I suggest turning toward the less cluttered side. You can use the remaining rays on each side to determine which side that is (count the number of hits in each side using the central ray as the mid point).
- Roam around
  - You can use the approach we used in class (a parallel composite with translation and rotation nodes) but to have a better control in this case I suggest to use a single goal for that. One possible approach is to always “mf” and then chose from time to time a turn action (“tr”, “tl”, “nt”) giving more probability to “nt” to avoid having a too erratic agent. Change the turn action for example only every 1 or 2 seconds.
- Facing the flower
  - One important basic behaviour is to center the agent toward the flower once it is detected. Again, it is not necessary to overcomplicate using angles or distances. First decide which direction to turn using the index of the ray that is detecting it. Then turn until you have the flower in the central ray.
- Get the flower
  - Once you have the flower in front of you, you only have to advance to get it. Be careful that, although the normal scenario is that by advancing the agent will get the flower, there is another possibility. In a scenario with another astronaut, you could detect a flower and advance to get it but the other Astronaut be quicker. If the flower disappears and the agent is not checking that the flower is still there, it will advance forever. You have to consider the non successful alternatives.

- Return to Base
  - I have detected that sometimes the “walk_to” action fails. You can use “teleport_to” instead.
  - You can use self.i_state.currentNamedLoc to check if the agent has arrived to the location “Base”.
  - Remember that, in that location (“Base”), the agent has the container at range and therefore is ready to unload the flowers.
- Unload flowers
  - The agent cannot carry more than two flowers, so once it has collected 2 flowers has to return to the “Base” and unload the flowers into the container. The action to unload the flowers is “leave,AlienFlower,2”.

- Final considerations
  - Start with a minimalist first version, focusing on simple, functional solutions for your goals. Avoid unnecessary complexity at this stage. Once the behaviour tree is working, you can optionally refine your goals with more sophisticated techniques—if time and interest permit. It is better to have a fully functional simple solution than two or three very sophisticated goals but no behabior tree.