PRACTICAL ASSIGNMENT #3

Behaviour Trees, mainly

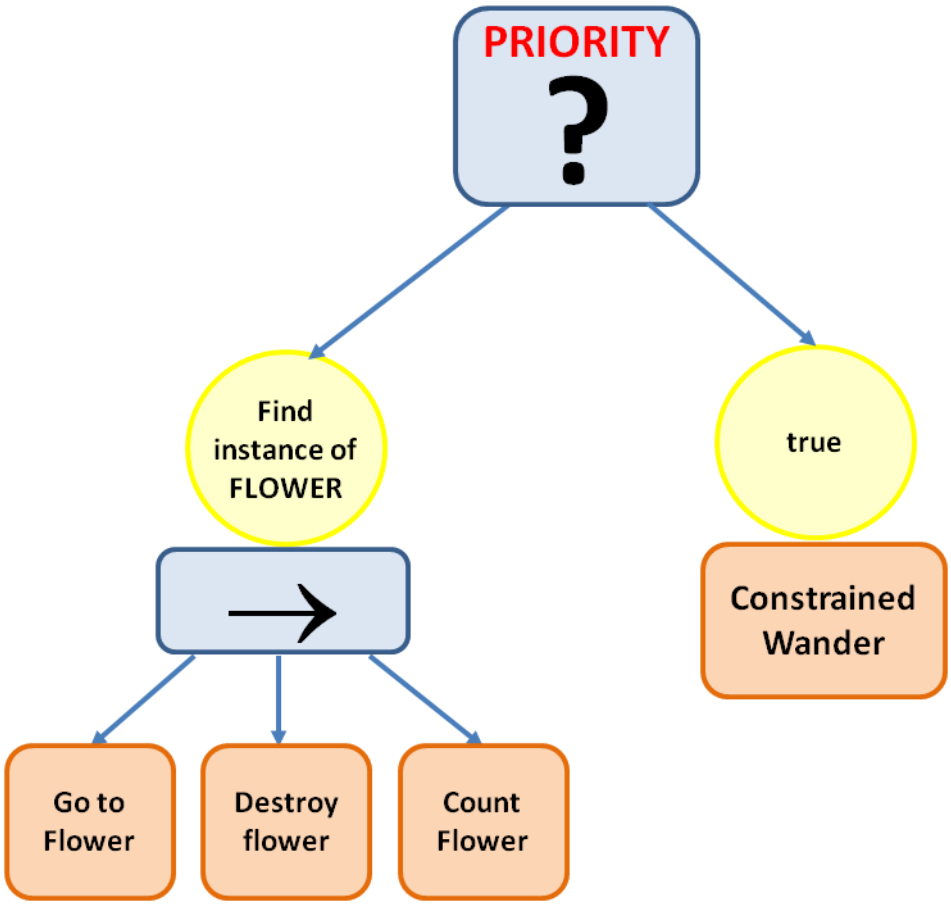
BOB's in love!!!

Bob, one of the main characters of the examples on behaviour trees is in love. He cannot take Daisy out of his (mainly bt-based) mind. He has made up his mind and today is going to be the day: he is going to ask Daisy to marry him. To do so, he needs to get a nice bouquet, go to her place and… But we all know Bob: he has not a lot of money and the little money he has, he spends on beer. Thus going to a flower shop is out of the question. He has decided to go to the nearest park and pick some flowers there. Ah, picking flowers under the sun! Isn't that a hard job that is likely to get him thirsty? And when Bob is thirsty...

PART 1 (COMPULSORY): MAX 70%

STEP 1: give Bob the ability to pick flowers

Create the following behaviour that will make Bob wander until he finds a flower…



Notice that as **constrained wander** never succeeds, this behaviour only succeeds when Bob has found a flower, and picked it. The flower is "destroyed" to make it disappear from the

scene. The function that you use to find an instance of FLOWER should keep returning true once an instance has been found (or, else, it could return false straight from the moment the flower is destroyed, not letting the sequencer terminate successfully –even avoiding the execution of the "count flower" action).

If now you "decorate" this tree with a "repeat forever" decorator, you'll have an "endlessly wander and pick flowers" behaviour. Name it **BT_PickFlowers**. Do not forget that this behaviour never ends.

STEP 2: behaviour trees depend on tasks (actions) and nothing prevents tasks being based on mechanisms other that BTs. Create an **FSM-based** behaviour that implements Bob's working routine: going to the boxes area, picking a box, transporting the box to the warehouse, going to the boxes area… This FSM:

- Enters a TERMINATED state once 3 boxes have been transported
- Enters a FAILED state if BOB runs out of boxes before 3 have been transported

Now you can code an action based on this FSM. This action starts the FSM and terminates when the FSM reaches the TERMINATED state, fails if the FSM reaches the FAILED state and has a status of running in any other case.

STEP 3: Using the work action coded in the previous step, you can "rewrite" BOB's **BT_QuenchThirst** behaviour (deciding whether to go to the bar or to the bank, once in the bank withdrawing money or going to work … it's the same behaviour shown in class)

STEP 4: picking flowers and thirst quenching, all in a single behaviour tree

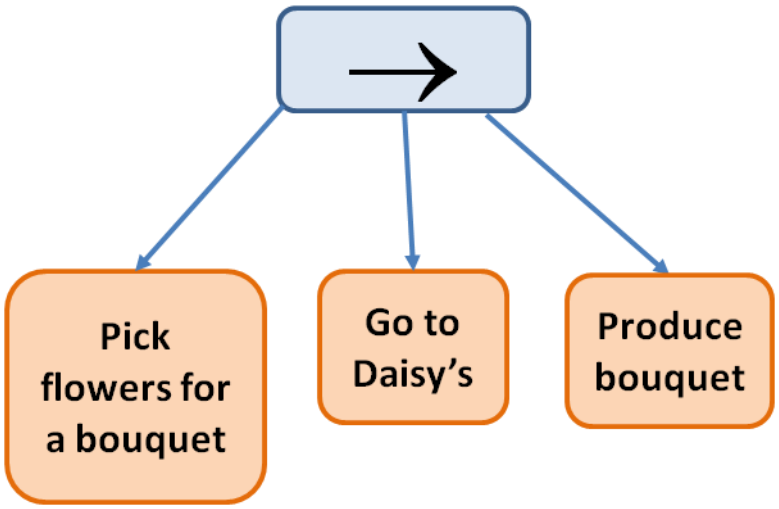
Now create another non-stop behaviour that engages Bob in a routine of picking flowers (the behaviour of the first stpe) that is interrupted every time his thirst level is too high. Being too thirsty makes BOB activate his "quench thirst" behaviour. You'll need a "too thirsty condition". Name this new behaviour **BT_PickAndQuench**.

STEP 5: Give Bob the ability to pick flowers (and quench his thirst) until he can make a bouquet

Now it's time to give Bob the behaviour that'll make him pick flowers (and quench his thirst when necessary) until he has got enough of them to make a nice bouquet. Build a new behaviour, based on the preceding one, that makes Bob pick flowers until he has reached the right number of them (no less than 30). You'll need a "hasEnoughFlowers" condition and a new field in Bob's blackboard. Name this behaviour **BT_PickForBouquet**. Notice that **this is not** a non-stop behaviour: it succeeds when the agent has got enough flowers.

STEP 6: Bob's final behaviour

Finally you have all the ingredients to concoct the recipe for Bob's behaviour:

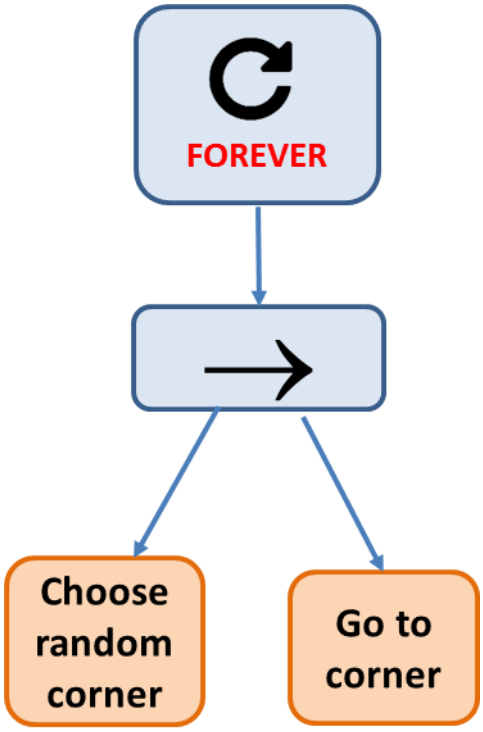


Obviously, the "Pick flowers for a bouquet" is the complex task performed by the behaviour tree developed in the previous stage. Name this final behaviour **BT_BOB**.

… And Daisy is waiting for a "formal" declaration

Daisy knows that Bob is in love with her (they always know, don't they?) and nervously waits for him to come to her place and ask her to marry him. That "being nervous" translates into pacing the house, going from corner to corner...

STEP 7: Daisy paces the house…



To build this behaviour tree you will need a "choose random corner action" (use waypoint objects to mark location of corners …). Name this behaviour **BT_WanderNervously**.

STEP 8: Daisy's final behaviour

Use the previous behaviour to build Daisy's final behaviour. Using a "priority selector", make Daisy nervously wander until a bouquet "appears" (findinstance…) near her (Bob will make it appear…). Then she walks towards the bouquet and accepts Bob's proposal giving him her heart. Name this behaviour **BT_Daisy**.

PART 2 (OPTIONAL): MAX +30%

[Solve this part only if the previous one is fully functional and error free.]

Add more.
Add some more complexity to BOBs behaviour. A good "starting point" could be the moment when the **work** behaviour fails because there are no more boxes to move. You can even consider the possibility of introducing new characters in the scene…

Write a short report explaining what are the extras you have added to the scene. Clearly specify new behaviours (diagrams of BTs, FSMs, specifications of new tasks, …) Without this report, this part will not be taken into account.

SUBMISSION

Due date: check eCampus

Instructions:
Deliver a one-scene or two-scene project
A project is provided as a starting point
Rename folder containing the project
Remember the required report for the optional part
Compress and deliver through eCampus (do not use e-mail, Dropbox, Google drive, …)