Exercise – Decision Trees

The goal of this tutorial is to implement a basic decision tree structure. This tutorial will make use of your existing Agents and their steering behaviours from the previous tutorials. We will be creating a basic decision tree structure and implementing decision actions that execute steering behaviours. Your agents will use the tree to decide which steering behaviour should be run. You will need Wander and Flee behaviours implemented.

Exercise 1:

We will be creating a basic Decision Tree for our agents so that they can make a decision if they should continue wandering or flee. To do so we will need to implement a base *Decision* class to derive our decisions off. Use the lecture's pseudo-code as a guide:

```
//base class for decisions
class Decision
    Agent owner
    func makeDecision() = 0
```

We will need a *WanderDecision* and a *FleeDecision* that each make use of a steering behaviours to control the owner:

```
//a decision tree node which performs a wander action
class WanderAction : Decision
    func makeDecision():
        //implements a wander behaviour

// a decision tree node which performs a flee action
class FleeAction : Decision
    func makeDecision():
        //implements a flee behaviour
```

To be able to make use of these ‘action’ decisions we need to come up with some branch decisions: Are we close to an enemy? Have we been fleeing for long enough?

One way to implement the behaviour we want is have an extra decision set a timer on the owner agent when we get close to the target agent. If the timer is set then the owner will make the decision to flee. If the timer is not set then the owner will make the decision to wander. Make sure that your agents either cannot wander too far away from each other or they wrap around when they go off the screen.

Our agents will need a timer and a Decision:

```
//AI agent from previous tutorials
class Agent
    Agent target            //other agent, used by some decisions
    Decision decision       //root decision for agent
    float timer             //used by some decisions

    func update():
        timer -= deltaTime
        decision.makeDecision()
```

The following could be the extra decisions needed for the decision tree to function:

```
//a decision tree node which sets the agent's 'timer' to 'length'
class SetTimerAction : Decision
    float length

    func makeDecision():
        owner.timer = length

//a decision tree node which branches to another decision based on 'timer'
class CheckTimerDecision : Decision
    Decision trueDecision        //run this if condition is true
    Decision falseDecision       //run this if condition is false

    func makeDecision():
        if owner.timer > 0
            trueDecision.makeDecision()
        else
            falseDecision.makeDecision()

//wander decision deriving from the decision base class
class CheckDistanceDecision : Decision
    Decision trueDecision        //run this if condition is true
    Decision falseDecision       //run this if condition is false
    float distance               //run 'trueDecision' if within this distance

    func makeDecision():
        if (owner.pos - owner.target.pos).magnitude < distance
            trueDecision.makeDecision()
        else
            falseDecision.makeDecision()
```

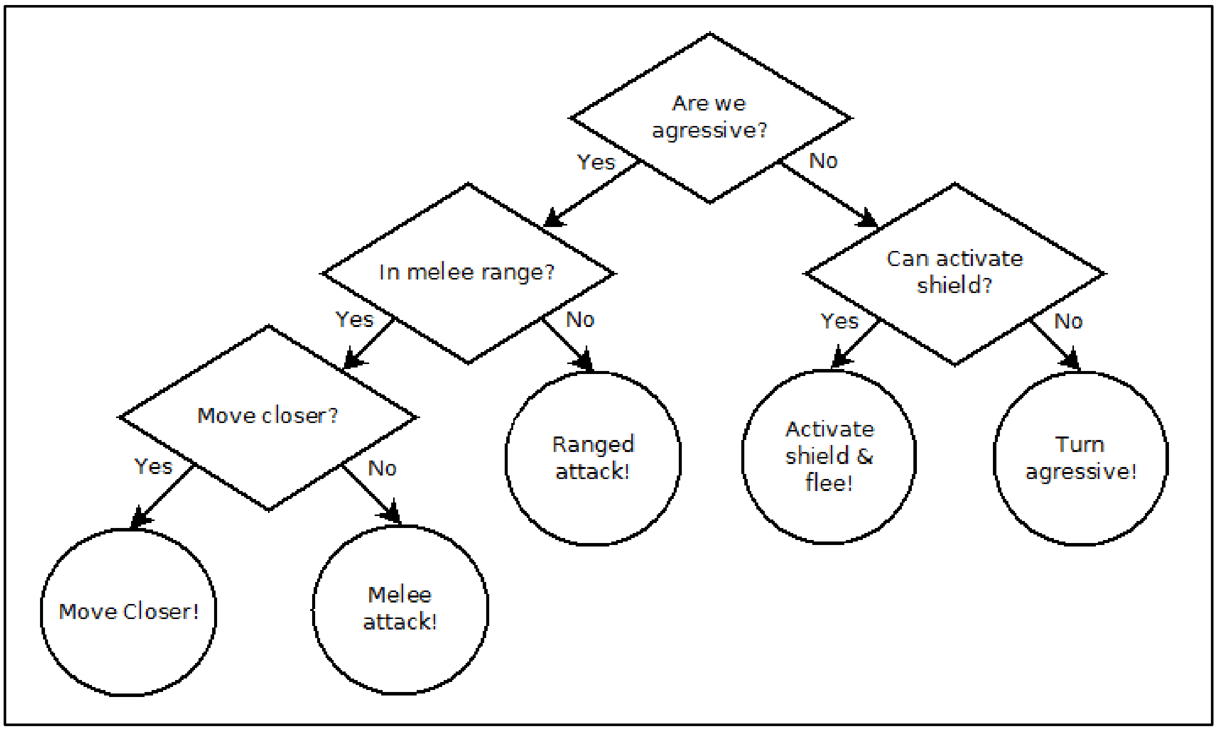
With the previous decisions setup, how could you design a tree to handle the following behaviour?

1. If the flee timer is greater than 0 then flee from the other agent.
2. If the distance to the other agent is less than a certain value then set the flee timer.
3. If neither of the previous statements are true then wander.

Try drawing your tree on paper first before you implement it in code.

Challenge Exercise:

Implement the following Decision Tree structure into your application for your agents to execute. Remember, you need only give the agent the root Decision Tree node. See if you can improve the decision tree by adding more decisions/actions.



The agent should have a shield that regenerates.

References:

- A great gamedev.stackexchange post on Decision Trees can be found [here].