3 Intelligent Systems Lab Assignment — Agent Coordination

(This lab assignment was redesigned last year by my two teaching assistants. They tested it much more than I usually test my own assignments.)

In this lab you will learn how agent performance can be influenced by the agent's own behaviour, but also through the behaviour of others. You will do this in a predator-prey chase scenario. Your agents will control a small pack of wolves (3) that are chasing a random moving prey. The prey is too large to be captured by a lone wolf, so two wolves need to be adjacent to the prey to be able to capture it.

The prey and wolves move around on a tiled torus (just like PacMan). Prey and wolves move every tick of the clock. The prey as implemented moves randomly. The movement of the wolves will have to be defined by you.

To define your wolves: implement the Wolf-interface. Depending on whether you want to limit your wolves' movement or not, you should implement either the moveAll or moveLim methods. Both methods receive two parameters. The first parameter is a list of all wolves, each represented by relative row and column distance to the wolf receiving the list. The second is a similar list, but for prey. The wolf can see the location of all other wolves, but it is not certain about the identity of the other wolves. Furthermore, it can only see prey within a Manhattan distance of 5 around it.

The method "moveAll" should return an array of length 2 with the first element indicating whether the wolf should move one row up or down or stay in the current row. Going up is represented by the value -1, going down with 1 ... you should be able to guess what the value for staying put is. The second element in the array will define the movement for the column dimension: left = -1, right = 1, not moving = 0. The method "moveLim" should return an integer in 0,1,2,3,4, where 0 means no movement, 1 = up, 2 = right, 3 = down and 4 = left. Movement that makes the wolf move into something on the grid (prey or other wolf) is not executed.

You can play around with the environment. By default you specify 5 wolves-types of which 3 are chosen randomly (without repetition) to chase some prey. Initial positions are also random. For debugging purposes, you can change the code so you only give three wolf-types that are all chosen.

Try to build the best wolf-team you can, and play around with the options you have. E.g.:

1. Building a homogeneous team vs a heterogeneous team: what is easiest? What is the best? What is the most robust?
2. Can you deal with a RandomWolf in your pack?
3. Does your strategy work with limited movement?
4. What happens if you increase or decrease the number of wolves?
5. What happens if you need more wolves to catch a single prey?
6. ...

To define clever wolves you should try to let them react to each other. Your wolves cannot communicate. There is no global control. Each wolf must behave based on what they see in their surroundings. There might be a very simple solution in which the wolves ignore each other, but it will probably take the wolves longer to catch the prey like that. A good first step is to define a following behaviour. From there on, be clever. And let your wolves be, too.

Handing in

Those of you who choose to earn their daily work points through the lab assignments: please upload your agent and a short report (in PDF format only, unzipped) through the Student Portal before the deadline if you want to get credit for your work. The report should include both a description of the Wolves you built, but also a selection of test outcomes. Also discuss what you observed and learned and how you approached the assignment, what challenges your Wolves faced etc. For those of you not wanting to get graded on this assignment, save your wolves and upload the code only to the student portal.