448.057 Context-Aware Computing (UE)
Exercise 1 – Termites gathering woodchips

Task description

The first system to be modelled is biologically inspired. As in many complex systems, a set of simple local rules can produce a high-level behaviour that has non-intuitive characteristics. An example of such a system is termites gathering woodchips. The termites follow a set of simple rules: each termite starts wandering randomly and, if it bumps into a woodchip, it picks the chip up and continues to wander randomly. When it bumps into another woodchip, it finds a nearby empty space and puts its chip down. With these simple rules, the woodchips eventually end-up in a single pile.

The task consists into building a NetLogo model that represents the behaviour of termites gathering woodchips into piles, according to the following specifications. The `setup` procedure must create the virtual world consisting of grassland, by spreading around woodchips, termites, and obstacles. These three operations are executed inside the sub-procedures `setup-grassland`, `setup-chips`, `setup-termites` and `setup-obstacles`, respectively, as shown in Listing 1.

Listing 1: Skeleton for the setup procedure.

```
to setup
        clear-all
        setup-grassland
        setup-obstacles
        setup-chips
        setup-termites
        reset-ticks
end
```

The virtual world is a torus (i.e., world-wrap is enabled) with a size of 90x90. The terrain of the virtual world is mostly grassland (modelled as green), but there are a number of obstacles of circular shape (modelled as black). The number of obstacles in the virtual world is an input variable `obstacles-number` and its range is $[0-3]$. Each obstacle in the virtual world has a radius that is randomly selected between 1 and `obstacles-radius`, with the latter being an input variable in the range $[2-15]$ (default is 12).

On the grassland, there are randomly spread woodchips modelled as brown. The percentage of grassland covered by woodchips depends on an input parameter called `density`, which can be a value between 1% and 40% (default is 20%). In the virtual world there can be a variable number of termites moving around. Their value is taken from an input variable `termites-number` and its range is $[1-100]$ (default is 35). Termites are red, have a size of 2 and their turtle shape should be selected accordingly. A termite that is carrying a woodchip has to be coloured with a different color, so to be recognizable. Termites are initially randomly spread over the green grassland.

The `go` procedure defines the local rules of the termites. The latter follow a set of three simple rules, as shown in Listing 2. Firstly, each termite has to pick up a chip (through the `pick-up-chip` procedure).

Towards this goal, each termite moves in a random direction (termites *cannot* walk on obstacles) until it travels over a wood-chip.

Listing 2: Skeleton for the go procedure.

```
to go
        ask turtles [ pick-up-chip ]
        ask turtles [ find-new-pile ]
        ask turtles [ drop-off-chip ]
        tick
end
```

However, if a termite has not found a wood chip after `tiredness` moves, it will stop and rest until the next simulation round. The `tiredness` is an input value between 1 and 100 (default is 100). At the end of the `pick-up-chip` procedure, a termite is therefore either sleeping or carrying a woodchip. The status of the termite has to be stored in the `carrying-chip` Boolean variable. Secondly, a termite which is carrying a woodchip has to move in a random direction until it finds a new pile where to place the woodchip. This is done in the procedure `find-new-pile`, and termites that are carrying a woodchip do not get tired. Thirdly, a termite has to move until it finds the first empty patch where to drop the woodchip (`drop-off-chip` procedure). Woodchips can only be deposited on grassland (i.e., not on any obstacle). After dropping off the chip, the termite has to move away from that area by `moving-away` units. The latter is a number taken from input in the range $[1 - 100]$ with default value 20.

All input variables have to be taken through NetLogo's slider widgets. In order to visualize the characteristics of the system in real-time, two plots have to be included in the model as well. A first plot named `woodchip-clustering` shows the woodchip clustering level (i.e., the number of woodchips completely surrounded by other woodchips in all adjacent patches). A second plot named `carrying-woodchips` shows the number of termites carrying woodchips (i.e., the number of termites that are not sleeping).

Finally, also four monitors have to be implemented, showing: (i) the amount of woodchips on the virtual world; (ii) the number of grassland on the virtual world; (iii) the amount of woodchips that are isolated from any other woodchip (i.e., the number of woodchips that are only surrounded by grass); (iv) the number of termites that are wandering around without carrying any woodchip should be implemented.

Basic functionality *(5.5 points)*. Create a NetLogo model that follows the specifications above. Code should be properly commented, and can be inspired by the termites' biology model. Each team member should be able to fully explain the submitted code.

System behaviour *(2.5 points)*. Discuss the high-level behaviour of the system by answering the following questions (please respect the space constraints):

1. Does the density of the woodchips affect the final shape of the wood piles?

Answer:

2. Is the global behaviour of the system influenced by the `tiredness` variable? If so, how?

   Answer:

3. What happens when the `moving-away` parameter is decreased? What causes this behaviour?

   Answer:

4. Edit your program in such a way that it does not allow termites to pick up a wood-chip that is completely surrounded by other woodchips (i.e., a termite will not remove woodchips that are already in the middle of a pile). Does this affect the overall behaviour of the system? If yes, how?

   Answer:

Extended functionality: vertical piles *(1 point)*. Extend your program by introducing the possibility to have several woodchips piled vertically on the same patch (e.g., a darker brown color can indicate that there are more woodchips on that patch). No more than 7 woodchips can be piled on top of each other.

Extended functionality: obstacle removal *(1 point)*. Extend your program in such a way, that termites remove a patch of obstacle and turn it into a woodchip at a price of increased tiredness. Add a slider (`obstacle-cost`) in the range $[2 - 10]$ (default value is 4). Whenever a termite stands next to an obstacle and is *not* carrying a woodchip it has to remove a patch of obstacle and turn it into a woodchip.

Please submit the filled PDF and your NetLogo source code to the TeachCenter as illustrated in the lecture within **Sunday, 04.11.2018, 23:59 CET**. Please submit the extended solution as separate file or use a switch to enable or disable the extended functionality.

Group ID:	
Stud. Names:	

Finalize Form