Handout

Using STARLOGO

Michael Muskulus* (University of Leiden)

1) Getting started

The interactive modelling environment STARLOGO is freely available from

`http://education.mit.edu/starlogo/`

Installation is straightforward. After launching STARLOGO, you will see the Control Center which is used for entering your commands and functions, and the so called Interface, which is used for graphical output and interactive control via buttons and monitors.

STARLOGO is different from other programming languages/environments. It is better not to think of a STARLOGO program as a usual program in which the control flows from one instruction to the next, but more as a world in which different objects exist and interact in certain ways.

Tbere are three basic objects in STARLOGO :

1. The observer is used to issue global commands. It can be thought of as being the origin of every 'program', using the other objects (by asking them to do things) as it is necessary for the purposes of the task at hand.

2. The grid is a rectangular area of 50x50 patches[1] on which the action takes place. This is the world, in which the observer and the turtles live.

3. The turtles are the basic mobile units living on the grid. They can walk around, draw lines and color their patches. They can also have different states and act according to them. The observer can destroy and create turtles as he needs them and ask them to do things.

*email: *muskulus@math.leidenuniv.nl*

[1] In fact, it should be possible to change the number of patches. Unfortunately, I have not found out yet how to do this! Please help me if you can!!

When writing a STARLOGO 'program', the first thing one has to do is to decide which objects are responsible for which (sub–)task. The turtles are the basic units, so for each action of the turtles, one should write a procedure (subroutine) that tells the turtles what to do. These procedures will be called by the observer when necessary and apply to all turtles at the same time. (It is also possible to pick out certain turtles, e.g. by considering some of their properties, but usually commands given to turtles apply to all of them in parallel).

2) Example: 2d Random Walk

I will demonstrate some of the capabilities of STARLOGO with an example: the 2d Random Walk. We are interested in the mean distance from the origin and its standard deviation. How do we tell STARLOGO to solve this task? Our first decision is about the different objects: We will use the turtles as our particles in the random walk. So each turtle will walk randomly over the grid, and we will write a procedure `move` which is used to move the turtles one step in a randomly chosen direction.

The code for this has to be entered in the "Turtle Procedures" window in the Control Center and looks as follows:

```
to move
seth pick [0 90 180 270]
fd 1
end
```

The first line begins with `to` which tells STARLOGO that you want to define a procedure.

In the next line, `pick` choses one element (number) randomly from the list of given values, which are angles. Then, `seth` ("set heading") is used to turn the turtle into the direction given by `pick`, so it will face northwards (0 degree), eastwards (90 degree), southwards (180 degree) or westwards (270 degree). Remember: Each turtle will do the same, but each with its own `pick`, so each turtle will face into a random direction, independently of the other turtles, afterwards! Next, we move the turtle one step forward with `fd` ("forward"). The definition of the procedure ends with `end`.

Now that we have defined the basic movement of the turtles, we will write an observer procedure `rw` (for "Random Walk") which will create turtles and let them walk over the grid for a certain time. There will also be some output. Let's look at the example program. It has to be typed into the "Observer Procedure" window of the Command Center:

```
to rw :nt :ns
ct
co
cg
crt :nt
```

```
ask-turtles [pd]
type "No. turtles: "
print :nt
type "No. steps: "
print :ns
repeat :ns [ask-turtles [move]]
print "Average Distance from Origin:"
type "  "
print average-of-turtles [distance 0 0]
print "Std-Dev Distance from Origin:"
type "  "
print sdev-of-turtles [distance 0 0]
end
```

The first line defines the procedure `rw` with two variables `:nt` ("number of turtles") and `:ns` ("number of steps"). These numbers will be used in the following. In the next lines, we destroy all turtles with `ct` ("clear turtles"), we also clear the textual and the graphical output with `co` ("clear output") and `cg` ("clear graphics").
With `crt` we create a number of turtles. These are then asked to put their pens down with `pd`, so that they will draw their trajectories when walking over the grid.
The commands `type` and `print` are used to write output into the output window (Question: Can you guess what the difference between them is?). At the start of STARLOGO, this window is usually not visible. Use the menu "Windows→Output Window" to make it appear on the screen.
The most important line is the one where we ask the turtles to move a step. This has to be repeated a (given) number of times. After that, we will output some statistics in the output window. Fortunately, STARLOGO knows many procedures which make this an easy task: With `distance 0 0` we get the distance for each turtle from its position to the origin (0 0). If we are interested in the average value over all turtles, the procedure `average-of-turtles` can be used. It needs a list of all the values as input, so we use brackets `[` and `]` to provide this. It is also possible to get the standard deviation (or variance, median, ...) with `sdev-of-turtles` (and respectively `variance-of-turtles`, ...).

Please try this program out! It can be run by typing something like `rw 10 10` in the "Observer Command Center" window in the Control Center. Of course, we might use more turtles to get a more accurate result: `rw 100 10`, or we might let the turtles walk for a longer time: `rw 10 100`.

I hope, that now you have become curious about the possibilities of STARLOGO . On the webpage given, you can find the whole documentation, covering all of the around 600 (!) commands that are available.

3) Suggestions and Exercises

Please try to familiarize yourself with the STARLOGO environment. It can be easily used to simulate a number of independent agents moving on a grid. For other tasks

of course, it is not so convenient (how would you do a Monte–Carlo integration with turtles?).

Exercise: Modify the Random Walk example program so that now each step can be in a random direction chosen uniformly from all of the 360 possible angles. Hint: Look up the command `random` in the documentation.

Exercise: Modify the Random Walk example program so that now there is a "trap" on the grid: A certain patch (gridbox), let's say the box with coordinates 10 10, is such that if a turtle enters it, the turtle will be killed. How much time does it take (on the average) for this to happen? Hint: Look up the commands `ask-patch-at`, `setpc`, `die`, `nobody`, `one-of-turtles`, `average-of-list` and `globals`.

Exercise: Can you write a version of Conway's Game of Life that runs on the grid? You can use the graphical editor to create a given initial condition and then a observer procedure should be called that updates the grid a given number of steps. Hint: Look up the commands `patches-own`, `nsum4`, `ask-patches`. This program might be slow to run. Perhaps you find a clever way to make it faster?

Have fun!

M. Muskulus